(2)

Adaptive Set-Membership Identification in $\mathcal{O}(m)$ Time for Linear-in-Parameters Models

J.R. Deller, Jr. and S.F. Odeh[1]

Michigan State University
Department of Electrical Engineering / 260 EB
Control, Systems, and Signal Processing Group: Speech Processing Laboratory
East Lansing, MI 48824-1226

(517) 353-8840 deller@ee.msu.edu

Revision of April 3, 1992

EDICS Category: 5.2

Abstract

This paper describes some fundamental contributions to the theory and applicability of optimal bounding ellipsoid (OBE) algorithms for signal processing. All reported OBE algorithms are placed in a general framework which fruitfully demonstrates the relationship between the set-membership principles and least square error identification. Within this framework, flexible measures for adding explicit adaptation capability are formulated and demonstrated through simulation. Computational complexity analysis of OBE algorithms reveals that they are of $\mathcal{O}(m^2)$ complexity per data sample with m the number of parameters identified, in spite of their well-known propensity toward highly-selective updating. Two very different approaches are described for rendering a specific OBE algorithm, the set-membership weighted recursive least squares algorithm, of $\mathcal{O}(m)$ complexity. The first approach involves an algorithmic solution in which a suboptimal test for innovation is employed. The performance is demonstrated through simulation. The second method is an architectural approach in which complexity is reduced through parallel computation.

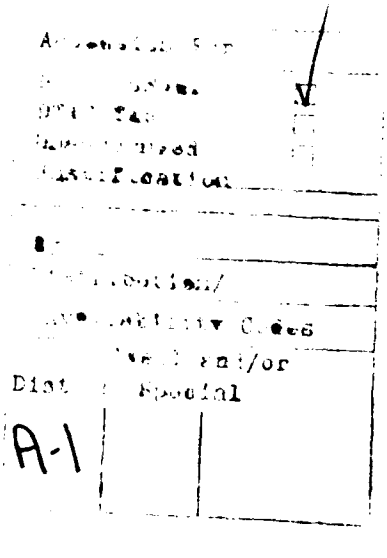
Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. MIP-9016734 and the Office of Naval Research under Grant No. N00014-91-J-1329. JD was also supported in part by an Ameritech Fellowship during the later period of this work.

The authors wish to thank several anonymous reviewers for their careful and constructive reviews. We are also indebted to Professors John P. Norton and Yih-Fang Huang for many helpful comments which significantly improved the paper.

[1]S.F. Odeh is currently with the Department of Electrical Engineering, University of Jordan, Amman.

List of Figures



List of Tables

List of Frequently Used Acronyms and Abbreviations

1. AR – autoregressive
2. ARX – autoregressive, exogenous input
3. BE – bounded error
4. LP – linear-in-parameters (model)
5. LSE – least square error
6. MIL-WRLS – WRLS algorithm based on matrix inversion lemma
7. OBE – optimal bounding ellipsoid (algorithm)
8. QR-WRLS – WRLS algorithm based on QR decomposition
9. RLS – recursive least squares (algorithm)
10. SM – set membership
11. SM-WRLS – set membership weighted recursive least squares (algorithm)
12. WRLS – weighted recursive least squares (algorithm)

1 Introduction

Set-membership (SM) identification of parametric systems is concerned with the computational description of *feasible sets* of solutions which are consistent with the measurements and the modelling assumptions. SM algorithms have been the subject of intense research effort in recent years and many approaches have been explored. The papers in [1] and [2] provide a broad and current overview of the area. In particular, comprehensive reviews of the field with extensive reference lists are found in papers by Walter and Piet-Lahanier [3] and by Milanese and Vicino [4]. An extensive list of application examples with references is also given in the Milanese paper. A tutorial on the principal algorithm of interest in this paper, the so-called *set-membership weighted recursive least squares (SM-WRLS)* algorithm, is found in [5].

One class of SM methods, the *optimal bounding ellipsoid (OBE)* algorithms[2], is of particular interest to the signal community since it represents a merging of the SM approach and widely used *least square error (LSE)* procedures for identifying linear models. The benefits of combining SM considerations (when they are known) with LSE processing are twofold: First, the SM information provides a feasible set of solutions which complements the unique LSE estimate. This feasible set can help to compensate for the restrictive nature of the assumptions placed upon the LSE model. Secondly, as we demonstrate in this paper, SM knowledge can greatly improve the efficiency of LSE identification.

Two aspects of OBE processing are treated in this paper. In a general way, it is shown that all reported OBE algorithms can be placed into a unified framework which is clearly related to conventional LSE processing. This framework will embrace *explicitly adaptive* OBE algorithms which will be demonstrated as a first major contribution of the paper. The second, and more extensive, aspect of this paper is concerned with the computational efficiency of OBE algorithms. OBE algorithms (both nonadaptive and adaptive) entail an interesting data selection procedure which typically discards 70 – 95% of the incoming data. The basis for this selective updating is a determination of whether the incoming datum is "informative" in the sense of refining the feasibility set. The selective updating procedure, however, generally does *not* imply a similar reduction in computational load, since the effort of checking for innovation in the data is approximately as expensive as the updating itself (the one exception to this rule is the OBE algorithm of Dasgupta and Huang [27] which is discussed below). Whether accepted or not, the processing requires $\mathcal{O}(m^2)$

[2]The original algorithm in this class due to Fogel and Huang [6] was called simply "OBE". We use this term to indicate the broader class of similar algorithms. The SM-WRLS algorithm will be seen below to be a specific type of OBE algorithm in this broader sense.

floating point operations[3] per incoming datum, where m represents the number of parameters to be estimated. Because much of the OBE research has focused on control systems applications in which m is relatively small, and where sample rates are low, this point has not been clearly brought out in the literature. However, as these algorithms gain wider acceptance in signal processing applications, computational efficiency will be more important. A second focus of this paper, therefore, is to demonstrate two very different methods for making a specific OBE algorithm run in $\mathcal{O}(m)$ time. The first solution is algorithmic, while the second is architectural. The ability to execute this interesting method in $\mathcal{O}(m)$ time makes it highly competitive with conventional identification techniques (especially *recursive least squares (RLS)*) which typically require $\mathcal{O}(m^2)$ flops per point.

2 An Adaptive SM-WRLS Algorithm

2.1 The Model and the LSE Identification Problem

The basic identification problem is as follows: We observe a system which is generating output sequence $y(\cdot)$ in response to input sequence $u(\cdot)$. Both input and output sequences are measurable, and $u(\cdot)$ is assumed to be a realization of a stationary, ergodic random process. The system is governed by a "true" model of form

$$y(n) = \boldsymbol{\theta}_*^T \boldsymbol{x}(n) + \varepsilon_*(n) \tag{1}$$

in which $\boldsymbol{x}(n)$ is some m-vector of functions of p lags of $y(\cdot)$ at time n, and q lags plus the present value of $u(\cdot)$, and where, $\varepsilon_*(\cdot)$ is the realization of a zero-mean, white noise error sequence. The error sequence is not measurable and the "true" parameters $\boldsymbol{\theta}_* \in \mathcal{R}^m$ are unknown. At time n we wish to use the observed data on $t \in [1, n]$ to deduce an estimated model which is similar in form to (1),

$$y(n) = \boldsymbol{\theta}^T(n) \boldsymbol{x}(n) + \varepsilon(n, \boldsymbol{\theta}(n)). \tag{2}$$

In the following, the identified parameter vector will be unique for each n (e.g. [7]), but will change at every step. Hence, we index the parameter estimate by n. The error sequence will depend on the choice of parameters, and we explicitly show this dependence. Neglecting the error term, this model exhibits only linear functional dependence upon the parameter vector and has been called a *linear in unknown coefficients* (e.g. [8]) or *linear-in-parameters (LP)* model (e.g. [3]). Special cases of the LP model of (2) are the *autoregressive-exogenous input (ARX)* and *autoregressive (AR)* models (e.g. [9] – [11]). For a current overview of methods that deal with nonlinear models, the reader is referred to [3],[4].

[3] One flop is taken to be a multiplication plus an addition operation.

In particular, we desire the weighted LSE model for which $\theta(n)$ minimizes $\xi(n) = \frac{1}{n}\sum_{t=1}^{n} \lambda_n(t)\varepsilon^2(t,\theta(n))$, where $\lambda_n(\cdot)$ is a sequence of nonnegative weights which may depend on n. $\theta(n)$ can be found as the solution of the following classical linear algebra problem (e.g. [7]): Given data (or a system of observations) on the interval $t \in [1,n]$ $(n \geq m)$, and some set of error minimization weights, say $\{\lambda_n(t),\ t = 1,2,\ldots,n\}$, form the overdetermined system of equations

$$\boldsymbol{X}(n)\boldsymbol{\nu} = \boldsymbol{y}(n)\ , \tag{3}$$

and find the LS estimate, $\boldsymbol{\theta}(n)$, for the vector $\boldsymbol{\nu}$. $\boldsymbol{X}(n)$ is the $m \times n$ matrix with i^{th} row $\sqrt{\lambda_n(i)}\boldsymbol{x}^T(i)$ and $\tilde{\boldsymbol{y}}(n)$ is the n-vector with i^{th} element $\sqrt{\lambda_n(i)}y(i)$. We will frequently refer to $(y(n),\boldsymbol{x}(n))$ as the *data set* at time n. The expression "per n" should be interpreted to mean "per data set."

In principle, the LSE solution is the solution to the *normal equations* (e.g. [7]), $\boldsymbol{C}(n)\boldsymbol{\theta}(n) = \boldsymbol{c}(n)$, where $\boldsymbol{C}(n)$ is the weighted *normal* matrix[4] [8, p. 62]

$$\boldsymbol{C}(n) = \boldsymbol{X}^T(n)\boldsymbol{X}(n) = \sum_{t=1}^{n} \lambda_n(t)\boldsymbol{x}(t)\boldsymbol{x}^T(t)\ . \tag{4}$$

and $\boldsymbol{c}(n) \stackrel{\text{def}}{=} \boldsymbol{X}^T(n)\boldsymbol{y}(n) = \sum_{t=1}^{n} \lambda_n(t)\boldsymbol{x}(t)y(t)$.

A recursive solution can be obtained for certain classes of time varying weights. Consider first the case in which the weights are time *invariant*, i.e. $\lambda_n(t)$ does not depend on n for any t. In this case, one can use a contemporary *weighted recursive least squares (WRLS)* algorithm based on the *QR decomposition* (e.g. [7]) of the $\boldsymbol{X}(n)$ matrix of (3). We shall refer to this algorithm as "QR-WRLS" to distinguish it from the more conventional WRLS algorithm based on the matrix inversion lemma (e.g. [8],[9] – [11]) (MIL-WRLS)[5]. QR-WRLS, in principle, involves the application of a sequence of orthogonal operators (Givens rotations) to (3) which leaves the system in the form

$$\begin{bmatrix} \boldsymbol{T}(n) \\ \hline \boldsymbol{0}_{(n-m)\times m} \end{bmatrix} \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{d}_1(n) \\ \hline \boldsymbol{d}_2(n) \end{bmatrix} \tag{5}$$

where the matrix $\boldsymbol{T}(n)$ is an $m \times m$ upper triangular Cholesky factor [7] of $\boldsymbol{C}(n)$, i.e., $\boldsymbol{C}(n) = \boldsymbol{X}^T(n)\boldsymbol{X}(n) = \boldsymbol{T}^T(n)\boldsymbol{T}(n)$, and $\boldsymbol{0}_{i\times j}$ denotes the $i \times j$ zero matrix. The system

$$\boldsymbol{T}(n)\boldsymbol{\theta}(n) = \boldsymbol{d}_1(n) \tag{6}$$

[4] In many contexts $C(n)$ is imprecisely called a "covariance" matrix. In fact, $\lim_{n\to\infty}(1/n)C(n)$ is the covariance matrix for the process if appropriate ergodicity assumptions are made.

[5] With the exception of the parallel processing architectures, developments throughout this paper may also be based upon MIL-WRLS. Indeed, almost all of the existing SM algorithms of the type considered here are based on the conventional method.

is easily solved using back substitution [7] to obtain the LSE estimate, $\boldsymbol{\theta}(n)$. This procedure can be performed in a recursive manner using only about m^2 memory locations. When the $n+1^{st}$ data set becomes available, it is weighted by $\sqrt{\lambda_n(n)}$ and incorporated into the system. Details are found in [12]–[14]. We shall use the name *QR-WRLS* to refer to this form of the recursion. It will be shown how this formulation makes possible the solution of the ellipsoid algorithms to be described on contemporary parallel architectures for great speed advantages. It also avoids initialization problems encountered in the use of MIL-WRLS [14].

The QR-WRLS algorithm can conveniently accommodate certain classes of time varying weights of interest in this work. The first is the case in which previous weights are *scaled* at time n by a time dependent scalar,

$$\lambda_n(t) = \frac{\lambda_{n-1}(t)}{\zeta(n-1)} \quad \forall t \leq n-1. \tag{7}$$

$\zeta(\cdot)$ is a *scaling sequence* which depends on the nature of the method. A common use for such scaling is to effect *adaptation by exponential forgetting.* In this case $\zeta(n) = \alpha^{-1}$, $\forall n$, where $0 < \alpha < 1$. This scaling is conveniently carried out in the course of QR-WRLS by simply multiplying the matrix and vector $\boldsymbol{T}(n)$ and $\boldsymbol{d}_1(n)$ by $\alpha^{-1/2}$ prior to considering $(y(n), \boldsymbol{x}(n))$ [13]. By a straightforward generalization of the work in [13], it can be shown that time-varying scaling may be accomplished by a similar premultiplication by $\zeta^{-1/2}(n-1)$. Let us denote the *scaled* system of equations at time $n-1$ by $\boldsymbol{T}_s(n-1)\boldsymbol{\theta}_s(n) = \boldsymbol{d}_{1,s}(n)$.

A second type of time varying weights is used to achieve *adaptation by exclusion.* In this case it is desired to remove some prior data sets from the system prior to considering $(y(n), \boldsymbol{x}(n))$. Let the set of times corresponding to data sets to be excluded be $\mathcal{T}_{n-1}$. Then, whereas $\lambda_{n-1}(t) > 0$, $t \in \mathcal{T}_{n-1}$, it is to be true that $\lambda_n(t) = 0$, $t \in \mathcal{T}_{n-1}$. This case is accomodated within QR-WRLS by simply reentering the data set to be forgotten with its previous weight as though it represented new data, then making some simple sign changes in the algorithm [5],[15]. Because the data sets are removed by "reversing" the Givens rotations which originally included them, this process is often call *back-rotation.* It is notable that previous data sets can likewise be *partially* excluded using a similar back-rotation method [16],[17]. After all desired data sets are removed, the system of equations is often said to be *downdated* at time $n-1$, and we shall denote this by writing

$$\boldsymbol{T}_d(n-1)\boldsymbol{\theta}_d(n) = \boldsymbol{d}_{1,d}(n). \tag{8}$$

If it were to be solved for, $\boldsymbol{\theta}_d(n)$ would represent an estimate at time $n-1$ without knowledge of the excluded data sets.

2.2 The BE Constraint and the Feasibility Set

A widely-research class of SM problems is those involving *bounded error (BE)* constraints (e.g. [3]–[6],[15]–[33]). In BE identification, a pointwise bound on the true error sequence is assumed. Ordinarily this takes the form[6]

$$\varepsilon_*^2(n) < \gamma(n), \tag{9}$$

where $\gamma(\cdot)$ is a known positive sequence. It follows immediately from (1) and (9) that the true parameters must be in the set

$$\omega(n) = \left\{\theta \mid \left(y(n) - \theta^T x(n)\right)^2 < \gamma(n)\right\}. \tag{10}$$

When intersected over a given time range, the sets $\omega(\cdot)$ usually form convex polytopes of feasible parameters, say $\Omega(n) = \bigcap_{t=1}^{n} \omega(t)$. Methods which track these polytopes [3],[4], [18]–[21] result in interesting but very complex algorithms which, at present, are not suitable for fast signal processing applications. OBE algorithms are of much lower complexity and work with an outer bounding hyperellipsoid, a superset of the polytope [6],[22]–[29]. The ellipsoid is "optimized" at each step by making some measure of its size as small as possible in light of the incoming data.

One of the drawbacks of the OBE approach from a set-theoretic point of view is that the hyperellipsoidal bounding sets are sometimes quite "loose" supersets of the actual feasibility sets (polytopes) (e.g. [22],[30]). This problem renders the resulting feasible superset "pessimistic" in that it may contain many points which are infeasible, and not reflect the size of the true feasible set. Whether certain measures can be taken, or particular OBE algorithms can be used, to minimize this problem, is an open issue. One possible solution is the use of *inner* bounds, as suggested in [30],[31]. In the present work the relative size of the bounding set will turn out to be somewhat inconsequential. It is the information afforded by the *existence* of the ellipsoid which is important.

2.3 Combining the BE and LSE Problems: The SM-WRLS Algorithm

OBE algorithms are fruitfully viewed as a marriage between the LSE and BE problems for LP models. With this point of view, signal processing engineers have begun to exploit the benefits of BE information in the context of LSE identification problems. In particular, LSE identifiers exploit no point-by-point information which can be used to ascertain the usefulness of observations. As a consequence, every point must be processed, and the entire parameter space is retained as a "feasible set." BE constraints, when they are known, provide a finite feasible set and offer the possibility of including only data points which contribute to the reduction of this set.

[6]This form is slightly less general than stating asymmetrical amplitude bounds, $\varepsilon_{\min}(n) < \varepsilon_*(n) < \varepsilon_{\max}(n)$, but the very slight loss of generality is worth the significant analytic gain afforded by this assumption.

As mentioned above, the polytope $\Omega(n)$ arising directly from BE considerations is not easy to track and manipulate. Further, $\Omega(n)$ is not clearly related to the LSE solution. However, it has been shown in three special cases of scaling sequences, $\zeta(\cdot)$ (recall definition below (7)) , that there is an outer bounding hyperellipsoid, say $\bar{\Omega}(n)$, which contains $\Omega(n)$ and which *is* closely associated with the LSE estimate $\boldsymbol{\theta}(n)$ [6],[26],[27]. A description of the hyperellipsoid is embodied in the following:

Proposition 1 *Let $\Omega(n) \subseteq \mathcal{R}^m$ be the feasibility set arising from BE constraints as above. Let $\boldsymbol{\theta}(n)$ denote the weighted LSE estimate with associated normal matrix $\boldsymbol{C}(n)$. The weights used in the estimation are $\lambda_n(\cdot)$ with $\lambda_n(1) > 0$. There exists a hyperellipsoidal set of parameter vectors, $\bar{\Omega}(n) \subseteq \mathcal{R}^m$, such that $\boldsymbol{\theta}_* \in \Omega(n) \subseteq \bar{\Omega}(n)$, which is given by*

$$\bar{\Omega}(n) = \left\{ \boldsymbol{\theta} \mid \left\{ [\boldsymbol{\theta} - \boldsymbol{\theta}(n)]^T \boldsymbol{\Phi}(n) [\boldsymbol{\theta} - \boldsymbol{\theta}(n)] \right\} < 1 \right\} \tag{11}$$

where $\kappa(n)$ is the scalar quantity, $\kappa(n) \stackrel{\text{def}}{=} \boldsymbol{\theta}^T(n)\boldsymbol{C}(n)\boldsymbol{\theta}(n) + \sum_{t=1}^{n} \gamma(n)\lambda_n(t) \left[1 - \gamma^{-1}(t)y^2(t)\right]$, and $\boldsymbol{\Phi}(n) = \boldsymbol{C}(n)/\kappa(n)$.

Note that the ellipsoid is centered on the LSE estimate, $\boldsymbol{\theta}(n)$, and its defining matrix is a scaled version of the normal matrix, $\boldsymbol{C}(n)$.

The proof of Proposition 1 is a generalization of the proofs of similar results for special cases (discussed below) found in [6] and [26]. Another related result for complex-valued, multiple input - multiple output systems is proved in [16],[34].

Clearly, the weights $\lambda_n(\cdot)$ parameterize $\bar{\Omega}(n)$ and determine its size and orientation in the parameter space. Because we want to work with *recursive* LSE estimation, in particular QR-WRLS, let us henceforth restrict our attention to weight sequences which conform to the simple forms of time variation described in Section 2.1 - scaling and exclusion. This effectively restricts to *one* the number of free parameters available to control the bounding ellipsoid. The central objective of an *optimal bounding ellipsoid (OBE)* algorithm is to employ these free weights in the context of LSE estimation to sequentially minimize the ellipsoid size in some sense. A significant benefit is that often no weight exists which minimizes the ellipsoid size in some sense, indicating that the incoming data set is uninformative in the SM sense.

In a general sense, reported (nonadaptive) OBE algorithms differ in the scaling sequences, $\zeta(\cdot)$, used in creating time varying weights. Fogel and Huang's original OBE algorithm (henceforth called *Fogel-Huang OBE*) [6], and the more recent method by Dasgupta and Huang (henceforth called *Dasgupta-Huang OBE*) [27], are not presented from this explicit LSE point of view, and this unified approach has not been widely discussed. Some general ideas along these lines may be inferred from [33] and a unified treatment will be found in [34]. The *set membership weighted recursive least*

squares (SM-WRLS) algorithm is the simplest in this sense, employing unity scaling, $\zeta(n) = 1\ \forall n$. We henceforth focus on SM-WRLS because this absence of scaling is essential to achieve the desired low complexity algorithm. Details of the other reported algorithms are left to the original papers and enhancements by Belforte *et al.* [22], and Rao *et al.* [23],[24].

Nonadaptive SM-WRLS (when based upon QR-WRLS) is comprised of the following steps: At time n,

1. In conjunction with the incoming data set $(y(n), \boldsymbol{x}(n))$, find the optimal weight, say $\lambda_n^*(n)$, which will (prospectively) minimize the size (according to some set measure) of $\bar{\Omega}(n)$, say $\mu\{\bar{\Omega}(n)\}$. (This will generally require knowledge of $\boldsymbol{C}(n-1)$ or $\boldsymbol{T}(n-1)$, and $\kappa(n-1)$.)

2. Discard the data set if $\lambda_n^*(n) \leq 0$.

3. Update $\boldsymbol{\theta}(n)$ using QR-WRLS (see Section 2.1).

4. Update $\kappa(n)$ of Proposition 1 according to

$$\kappa(n) = \| \boldsymbol{d}_1(n) \|^2 + \tilde{\kappa}(n) \tag{12}$$

with

$$\tilde{\kappa}(n) = \tilde{\kappa}(n-1) + \lambda_n(n)\gamma(n)\left(1 - \gamma^{-1}(n)y^2(n)\right) \tag{13}$$

where $\tilde{\kappa}(0) \stackrel{def}{=} 0$.

Expressions (12) and (13) are derived in [5],[15]. A detailed version of SM-WRLS is described in [5].

2.4 Adaptation by Back-Rotation

OBE algorithms in general, and the SM-WRLS algorithm in particular, have been shown in many simulations to have superior tracking capability for time varying systems when compared with conventional RLS and other "nonadaptive" identifiers. In some sense, this improved tracking is a result of inherent and fortuitous adaptive properties arising from the the optimal weighting strategies which induce "forgetting" by virtue of increasing weights, or through the scaling factors $\zeta(\cdot)$. This "adaptation" pertains to the central estimate $\boldsymbol{\theta}(\cdot)$ only, and there is no provision for adapting the feasibility set $\bar{\Omega}(\cdot)$ to time varying dynamics. Consequently, the tracking behavior of "conventional" OBE algorithms is not predictable nor controllable. However, measures have been suggested by Deller and Odeh [5],[15]–[17], and Norton and Mo [33] to render explicit and controllable adaptation. All adaptive strategies for ellipsoid algorithms work on the general principle of inflating the "current" ellipsoid in some sense before considering an incoming data set. The basis for this

inflation is to contain the shifting true parameters while at the same time increasing some measure of "size" of the ellipsoid (see (16) and (17) below), making it more likely that the incoming data, with potentially novel information, will be selected.

For SM-WRLS, simple forms of adaptation have been based upon exponential forgetting and by exclusion or back-rotation [5],[15]–[17]. Norton and Mo have also worked with exponential forgetting and other forms of adaptation in a broader context [33]. While exponential forgetting is conveniently integrated into OBE algorithms, in the following we shall focus exclusively upon adaptation methods which are based on back-rotation, for two reasons: First, exponential forgetting precludes the achievement of the low complexity algorithm ultimately sought in this work. Secondly, due to the fact that heavily weighted points remain influential in the estimate for very long periods of time, exponential forgetting has not been found to be as effective in tracking fast time variations in system dynamics [16],[34]. In the case of adaptation by back-rotation, the system of equations (6) is downdated prior to considering the data set at time n. The result is (8). The altered ellipsoid is centered on $\boldsymbol{\theta}_d(n-1)$ and has associated matrix $\boldsymbol{C}_d(n-1)/\kappa_d(n-1) = \boldsymbol{T}_d^T(n-1)\boldsymbol{T}_d(n-1)/\kappa_d(n-1)$.

Proper downdating of the scalar $\kappa(n-1)$ is easy. Upon rewriting the definition of $\kappa(\cdot)$ from Proposition 1 at time $n-1$,

$$\kappa(n-1) = \boldsymbol{\theta}^T(n-1)\boldsymbol{C}(n-1)\boldsymbol{\theta}(n-1) + \sum_{t=1}^{n-1} \lambda_{n-1}(t)\gamma(t)\left[-\gamma^{-1}(t)y^2(t)\right], \tag{14}$$

it becomes immediately clear that if data sets at times $t \in \mathcal{T}_{n-1}$ are eliminated from the system, then the normal matrix is simply replaced by its downdated version and all deleted terms should be removed from the sum on the right. Correspondingly, the downdated version of (12) written for time $n-1$ becomes

$$\kappa_d(n-1) = \| \boldsymbol{d}_{1,d}(n-1) \|^2 + \left[\tilde{\kappa}(n-1) - \sum_{\substack{t=1 \\ t\in\mathcal{T}_{n-1}}}^{n} \lambda_{n-1}(t)\gamma(t)\left(1-\gamma^{-1}(t)y^2(t)\right)\right] \tag{15}$$

and the term $\tilde{\kappa}(n-1)$ in (13) should be replaced by $\tilde{\kappa}_d(n-1)$ which is defined to be the term in square brackets.

A wide range of adaptation strategies is inherent in the general formulation described above, many of them computationally inexpensive. We have found two forms of adaptation by back-rotation to be particularly effective. These are:

1. *Windowing.* Let l be a fixed "window length." For each $n > l$, let $\mathcal{T}_{n-1} = \{n-l\}$.

2. *Selective Forgetting.* At time n check some predetermined criterion indicating whether adaptation is necessary. If so, select the set to be forgotten according to some other criterion.

The first case above corresponds to the use of a sliding rectangular window of length l, outside of which all previous data sets are completely removed. The estimate at time n covers the range $[n - l + 1, n]$. The windowing technique is made possible by the ability to completely and systematically remove data sets at the trailing edge of the window. Only one back-rotation is required prior to optimizing at time n, and this is only necessary if $\lambda^{*}_{n-l}(n-l) \neq 0$.

At significantly higher computational expense, smoother windows can be implemented by back-rotation. This is accomplished by partial rotation of an included data set according to a schedule which gradually eliminates the data set [16],[17]. Since each included data set is back-rotated many times, the computation required to effect such a window is frequently not warranted by the benefits of slightly improved frequency resolution. For details, see [16].

Selective forgetting represents a very general class of techniques in which the data sets to be removed from the system are selected according to certain user defined criteria. The selection process can be, for example, to remove (or downweight) only the previously heavily weighted data sets, to remove the data sets that were accepted in regions of abrupt change in the signal dynamics, or to remove the data sets starting from the first data set and proceeding sequentially. Whatever the criterion, a fundamental issue is to detect *when* adaptation is needed to improve the parameter estimates. An example is explored in the simulation studies below.

2.5 Optimization

In the nonadaptive case, Fogel and Huang [6] suggest two set measures on $\bar{\Omega}(n)$ for optimization. These measures may also be applied to the downdated system extant at time $n-1$ if adaptation is employed. For generality, we assume downdating in the following. If adaptation is not used, it is only necessary to drop subscripts "d" where they occur. The first Fogel and Huang set measure is the determinant of the matrix $\boldsymbol{\Phi}^{-1}(n)$,

$$\mu_v\{\bar{\Omega}(n)\} \stackrel{\text{def}}{=} \det\{\boldsymbol{\Phi}^{-1}(n)\} \tag{16}$$

and the second is the trace,

$$\mu_t\{\bar{\Omega}(n)\} \stackrel{\text{def}}{=} \text{tr}\,\{\boldsymbol{\Phi}^{-1}(n)\}. \tag{17}$$

$\mu_v\{\bar{\Omega}(n)\}$ is proportional to the square of the volume of $\bar{\Omega}(n)$ while $\mu_t\{\bar{\Omega}(n)\}$ is proportional to the sum of the squares of its semi-axes. The following is a slightly generalized version (to accommodate adaptation by downdating) of results found in [6],[26]. Further generalizations are found in [34].

Proposition 2 *Let $\mathcal{T}_{n-1}$ be the set of times corresponding to data sets to be excluded by back-rotation prior to time n. Then let $\lambda_n(t), t \in [1,n]$ indicate the weights to be used to optimize (16) or (17) at time n. Under the adaptation by exclusion policy, for $t \in [1, n-1]$ and $t \notin \mathcal{T}_{n-1}$, $\lambda_n(t) = \lambda_{n-1}(t)$. For $t \in [1, n-1]$ and $t \in \mathcal{T}_{n-1}$, $\lambda_n(t) = 0$. Then,*

1. *if it exists, $\lambda_n^*(n)$ which minimizes the volume measure (16) is the unique positive root of the quadratic equation*

$$F_v(\lambda) = a_2\lambda^2 + a_1\lambda + a_0 = 0 \tag{18}$$

where, $a_2 = \{(m-1)\gamma(n)G_d^2(n)\}$,

$a_1 = \{(2m-1) + \gamma^{-1}(n)\varepsilon^2(n, \boldsymbol{\theta}_d(n-1)) - \kappa_d(n-1)\gamma^{-1}(n)G_d(n)\}\,\gamma(n)G_d(n)$,

and $a_0 = m\,[\gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1))] - \kappa_d(n-1)G_d(n)$,

in which all quantities are defined above except $G_d(n) \stackrel{\text{def}}{=} \boldsymbol{x}^T(n)\boldsymbol{C}_d^{-1}(n)\boldsymbol{x}(n)$.

2. *if it exists, the weight $\lambda_n^*(n)$ which minimizes the trace measure (17), is the unique positive root of the cubic equation*

$$F_t(\lambda) = b_3\lambda^3 + b_2\lambda^2 + b_1\lambda + b_0 \tag{19}$$

with $b_3 = \gamma(n)G_d^2(G_d(n) - I_d(n-1)H_d(n))$,

$b_2 = 3\gamma(n)G_d(n)[G_d(n) - I_d(n-1)H_d(n)]$,

$b_1 = H_d(n)G_d(n)I_d(n-1)\kappa_d(n-1)$

$-2H_d(n)I_d(n-1)[\gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1))]$

$-G_d(n)\varepsilon^2(n, \boldsymbol{\theta}_d(n-1)) + 3\gamma(n)G_d(n)$,

and $b_0 = \gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1)) - H_d(n)I_d(n-1)\kappa_d(n-1)$,

where $H_d(n) \stackrel{\text{def}}{=} \boldsymbol{x}^T(n)\boldsymbol{C}_d^{-2}(n)x(n)$ and $I_d(n) \stackrel{\text{def}}{=}$ tr $\boldsymbol{C}_d(n)$.

For later computational considerations we note the following. In the context of QR-WRLS, the inverse normal matrix, $\boldsymbol{C}_d^{-1}(n-1)$, never appears, yet it is needed to compute $G_d(n)$ and $H_d(n)$. The following circumvents this problem:

Lemma 1 *In the context of QR-WRLS, the scalars $G_d(n)$ and $H_d(n)$ can be computed using $\mathcal{O}(m^2/2)$ flops.*

Proof: Write

$$G_d(n) = \boldsymbol{x}^T(n)\boldsymbol{T}_d^{-1}(n-1)\boldsymbol{T}^{-T}(n-1)\boldsymbol{x}(n) \stackrel{\text{def}}{=} \boldsymbol{g}^T(n)\boldsymbol{g}(n) = \| \boldsymbol{g}(n) \|^2 \tag{20}$$

in which $\| \cdot \|$ denotes the l_2 norm. Now $\boldsymbol{x}(n) = \boldsymbol{T}^T(n-1)\boldsymbol{g}(n)$, and $\boldsymbol{T}^T(n-1)$ is lower triangular, so $\boldsymbol{g}(n)$ is found by back-substitution using $(m^2+m)/2$ flops. Now note that

$$\begin{aligned} H_d(n) &= \boldsymbol{x}^T(n)\boldsymbol{T}_d^{-1}(n-1)\boldsymbol{T}^{-T}(n-1)\boldsymbol{T}_d^{-1}(n-1)\boldsymbol{T}_d^{-T}(n-1)\boldsymbol{x}(n) \\ &= \boldsymbol{g}^T(n)\boldsymbol{T}_d^{-1}(n-1)\boldsymbol{T}^{-T}(n-1)\boldsymbol{g}(n) \stackrel{\text{def}}{=} \boldsymbol{h}^T(n)\boldsymbol{h}(n) = \| \boldsymbol{h}(n) \|^2 \end{aligned} \tag{21}$$

and back-substitution can once again be employed. □

3 Implementing SM-WRLS in $\mathcal{O}(m)$ Time

3.1 Complexity Considerations

A precise comparison of the computational loads of various OBE algorithms is given in [34]. The number of flops (see footnote 3) required for the (generally adaptive) SM-WRLS algorithm under consideration here may be approximated by

$$f_{opt} \sim \mathcal{O}(c_1 m^2) + b\mathcal{O}(c_2 m^2) + \rho\mathcal{O}(c_3 m^2) \tag{22}$$

where, ρ is the average number of data sets accepted per n; b is the average number of back-rotations per n; and c_1, c_2 and c_3 are small numbers (all in the range 0.5 – 2.5) which depend upon whether QR-WRLS or MIL-WRLS is used. For QR-WRLS upon which we have principally focussed in this paper, $c_1 = 0.5$, $c_2 = 2$, and $c_3 = 2.5$. The first term is due to the procedure which checks for information in the incoming data. The others are attributable to adaptation, and solution update, respectively. If either an exponential forgetting factor or a non-unity scaling sequence (other OBE algorithms), is used, an additional term of $\mathcal{O}(m^2/2)$ must be added. Apparently, the SM-WRLS algorithm, as presently formulated, is an "$\mathcal{O}(m^2)$" process. The objective of the section below is to demonstrate two distinct methods for reducing the effective complexity to $\mathcal{O}(m)$, thereby making a SM-WRLS algorithm a desirable alternative to standard RLS-based methods from a computational point of view.

Two approaches are taken. The first is an algorithmic solution which will reduce the true complexity to $\mathcal{O}(m)$ for processing on a sequential machine. The second is a hardware solution which reduces the basic algorithm to $\mathcal{O}(m)$ *parallel* complexity, with even further reduction possible if the algorithmic measures are combined.

3.2 $\mathcal{O}(m)$ Processing on a Sequential Machine

From a signal processing point of view, one of the most interesting aspects of an OBE algorithm is its inherent ability to select only data points which are informative in the sense of refining the feasible set. The fact that typically 70 – 95% of the data are rejected by this criterion (e.g. [6],[17],[23]–[29]) would seem to imply a remarkable savings in computation. However, this is only true to the extent that the *checking* for usefulness of the incoming data set is negligibly expensive compared with the inclusion of it in the estimate. We have seen above, however, that the checking procedure is not inexpensive (see lead term of (22)) – a point which has not been made clear in reported research. The approach taken here is to render the checking procedure an $\mathcal{O}(m)$ process in a manner which does not (empirically) degrade performance of the algorithm.

Before detailing the methods, some points about the use of the approximation "$\mathcal{O}(m)$" are necessary. The first concerns a practical matter. The objective in the following is to reduce the computational complexity of the algorithms to an *average* of $\mathcal{O}(m)$ flops per n. It will be appreciated that, without data buffering, the data flow is still limited by the worst case $\mathcal{O}(m^2)$ computation. However, if a buffer is included, the algorithm easily be structured to operate in $\mathcal{O}(m)$ average time per n. Further, by using interrupt driven processing of the checking procedure, it may be possible to reduce the average time even further. Other points concern algorithmic details. We reiterate that the use of a unity scaling sequence (SM-WRLS algorithm) is required in order to avoid an invariant $\mathcal{O}(m^2/2)$ flops per n. We specifically assume the use of this algorithm below, although the $\mathcal{O}(m)$ checking procedure to be developed does not depend on this choice. Secondly, (22) indicates that an adaptive strategy must involve a sufficiently small average number of back-rotations per n so that the $\mathcal{O}(m^2)$ adaptation term in (22) does not overwhelm gains made by reducing the checking cost. In the windowing case above, for example, we would expect that $b \approx \rho$ and the adaptation is not unduly expensive. A selective forgetting strategy which meets this condition will also be illustrated in the simulations below. Finally, we note that even if the checking procedure can be made $\mathcal{O}(m)$, terms $b\mathcal{O}(m^2)$ and $\rho\mathcal{O}(m^2)$ (typically $b \approx \rho$) persist in (22). This means that to truly achieve $\mathcal{O}(m)$ complexity, b and ρ must be $\mathcal{O}(1/m)$. For large m, this will not be always be the case. In fact, some experimental evidence suggests, not unexpectedly, that ρ increases, rather than decreases, with increasing m. For "large" m (conservatively, say, $m > 10$), therefore, it is the case that the complexity is reduced to $\mathcal{O}(\rho m^2)$ by $\mathcal{O}(m)$ checking. It should be clear however, that neither $\mathcal{O}(m)$ nor $\mathcal{O}(\rho m^2)$ complexity can be achieved if the checking procedure remains $\mathcal{O}(m^2)$. We therefore pursue an $\mathcal{O}(m)$ test for information in an incoming data set.

In principle, the information checking procedure for the volume or trace algorithms consists of forming either $F_v(\lambda)$ or $F_t(\lambda)$ of (18) and (19), then solving for the positive root. However, since $a_2 > 0$, and $b_i > 0$, $i = 1, 2, 3$, there is at most one such root in either case, and the test reduces to one of checking the zero order coefficient for negativity [35]. When the test is successful, then the root solving and updating proceeds, requiring the standard MIL- or QR-WRLS load, plus a few operations for finding the optimal weight. In spite of Lemma 1, the most expensive aspect of this information test is the computation of the quantity $G_d(n)$ or $H_d(n)$, each requiring $\mathcal{O}(m^2/2)$ flops. The trick to making the SM-WRLS algorithm an $\mathcal{O}(m)$ procedure is to find a way to avoid the computation of $G_d(n)$ or $H_d(n)$ at each n. We first develop a method which accomplishes this for the "volume" algorithm, then argue that it pertains to the "trace" optimization criterion as well.

Let us denote the estimation *error* vector at time n by

$$\tilde{\boldsymbol{\theta}}(n) \stackrel{\text{def}}{=} \boldsymbol{\theta}_* - \boldsymbol{\theta}(n). \tag{23}$$

It follows immediately from (11) that $\tilde{\boldsymbol{\theta}}^T(n)\boldsymbol{C}(n)\tilde{\boldsymbol{\theta}}(n) < \kappa(n)$. While it is tempting to view $\kappa(n)$ as a bound on $\tilde{\boldsymbol{\theta}}(n)$ (see discussion of the Dasgupta–Huang algorithm below), it is important to note that each side of this inequality is dependent upon $\lambda_n(n)$. In fact, let us temporarily write the two key quantities as functions of $\lambda_n(n)$: $\boldsymbol{C}(n, \lambda_n(n))$ and $\kappa(n, \lambda_n(n))$ and consider the usual volume quantity to be minimized at time n,

$$\mu_v\{\bar{\Omega}(n)\} = \det\left[\kappa(n, \lambda_n(n))\boldsymbol{C}^{-1}(n, \lambda_n(n))\right]. \tag{24}$$

It is assumed that enough data sets have been included in the normal matrix at time $n-1$ so that its elements are large with respect to the data in the incoming data set. For the choice of weighting strategy employed here, the quantity $\det \boldsymbol{C}(n, \lambda_n(n))$ is readily shown to be monotonically increasing with respect to $\lambda_n(n)$ on the domain $(0, \infty)$ [16], with $\boldsymbol{C}(n, 0) \stackrel{\text{def}}{=} \boldsymbol{C}(n-1, \lambda^*_{n-1}(n-1))$. Under the assumption above, $\det \boldsymbol{C}(n, \lambda_n(n))$ will not increase significantly over reasonably small values of $\lambda_n(n)$. The attempt to maximize $\det \boldsymbol{C}(n, \lambda_n(n))$ in (24) causes a tendency to increase $\lambda_n(n)$ in the usual optimization process. However, the attempt to minimize $\kappa(n, \lambda_n(n))$ generally causes a tendency toward small values of $\lambda_n(n)$, unless a minimum of $\kappa(n, \lambda_n(n))$ occurs at a "large" value of $\lambda_n(n)$. To pursue this idea and further points of the argument, we use two key facts about $\kappa(n, \lambda(n))$:

Proposition 3 *$\kappa(n, \lambda_n(n))$ has the following properties: 1. On the interval $\lambda_n(n) \in (0, \infty)$, $\kappa(n, \lambda_n(n))$ is either monotonically increasing or it has a single minimum. 2. $\kappa(n, \lambda_n(n))$ has a minimum on $\lambda_n(n) \in (0, \infty)$ iff*

$$\varepsilon^2(n, \boldsymbol{\theta}_d(n-1)) > \gamma(n). \tag{25}$$

To verify this result we need the following which is proven in [34]:

Lemma 2 *For $n > 1$, the sequence $\kappa(\cdot)$ can be computed recursively as*

$$\kappa(n) = \kappa_d(n-1) + \lambda_n(n)\gamma(n) - \lambda_n(n)\frac{\varepsilon^2(n, \boldsymbol{\theta}_d(n-1))}{1 + \lambda_n(n)G_d(n)}. \tag{26}$$

Proof of Proposition 3: For simplicity, we write $\lambda_n(n)$ as λ. Using (26) from Lemma 2, we can write

$$Q(\lambda) \stackrel{\text{def}}{=} \frac{\partial \kappa(n, \lambda)}{\partial \lambda} = \frac{G_d^2(n)\gamma(n)\lambda^2 + 2G_d(n)\gamma(n)\lambda + [\gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1))]}{G_d(n)^2\lambda^2 + 2G_d(n)\lambda + 1} \tag{27}$$

and

$$\dot{Q}(\lambda) = \frac{\partial^2 \kappa(n, \lambda)}{\partial \lambda^2} = \frac{2[G_d^2(n) + \gamma(n)G_d(n)]\varepsilon^2(n, \boldsymbol{\theta}_d(n-1))}{(G_d^2(n)\lambda^2 + 2G_d(n)\lambda + 1)^2}. \tag{28}$$

The denominator of $Q(\lambda)$ is positive on $\lambda \in (0, \infty)$ and therefore has a root on $\lambda \in (0, \infty)$ iff its numerator does. The the numerator is a convex parabola with its minumum at $\lambda = -1/G_d(n) < 0$, and it therefore has a unique positive root on the interval $(0, \infty)$ iff $\gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1)) < 0$. Further $\dot{Q}(\lambda) > 0$ for all $\lambda > 0$, so the root, if it exists, will correspond to a minimum of $\kappa(n, \lambda)$.□

Accordingly, it can be argued that: If $\det \boldsymbol{C}(n, \lambda_n(n))$ is increasing, but not changing significantly over reasonably small values of $\lambda_n(n)$, then it is sufficient to seek $\lambda_n(n)$ which minimizes $\kappa(n, \lambda_n(n))$. If $\kappa(n, \lambda_n(n))$ is monotonically increasing on $\lambda_n(n) \geq 0$, this value is $\lambda_n(n) = 0$ which corresponds to rejection of the data set at time n. It suffices, therefore to have a test for a minimum of $\kappa(n, \lambda_n(n))$ on positive $\lambda_n(n)$. A simple test is embodied in condition (25) which determines whether the square of the current residual exceeds the upcoming error bound. If this test is met, it is then cost effective to proceed with the standard optimization centered on (18). Otherwise, the explicit construction and solution of a_0 of (18) can be avoided.

In fact, this suboptimal test for innovation is similar to that used in the Dasgupta–Huang OBE algorithm reported in [27]. The suboptimal test of Dasgupta and Huang is to accept the incoming data set only if[7] $\varepsilon^2(n, \boldsymbol{\theta}(n-1)) < \gamma(n) - \kappa(n-1)$. This inequality likewise tests for a minimum of κ with respect to $\lambda_n(n)$, and differs in form from (25) because of the scaling factors (see (7) and surrounding discussion) which depend on the optimal weights, $\zeta(n-1) = (1 - \lambda_n^*(n))^{-1}$ in the Dasgupta–Huang case. While this dependence precludes the construction of a reasonable expression in $\lambda_n^*(n)$ with which to minimize the set measure $\mu_v\{\bar{\Omega}(n)\}$, the Dasgupta–Huang hyperellipsoid nevertheless *does* have a volume at each n, and it is therefore possible to attempt to apply the above arguments. A problem arises in the Dasgupta–Huang case, however, because the relative independence of $\boldsymbol{C}(n)$ and $\lambda_n(n)$ is not tennably argued. Therefore, the simplified test in this case is not subject to the "same" justification as (25). Interestingly, however, if $\lambda_n(n)$, which is already constrained to $[0, 1)$ in the Dasgupta–Huang algorithm, happens to be very small at a particular n, then the algorithm approaches the case of unity scale factors ($\zeta(n) \approx 1$) as in SM-WRLS, and it can be argued that the normal matrix changes only slightly. In this case, but only in this case, the arguments above are applicable. Of course, artificially constraining the weights to be small for all n destroys the optimization process in the Dasgupta-Huang method, so that this analysis provides support for the suboptimal test only for isolated and infrequent times. Dasgupta and Huang argue simply that $\kappa(n)$ is "a bound on the estimation error," and should be minimized. This claim has been disputed by Norton and Mo [33] and is not clearly supported here. Generally, the arguments in support of (25) are valid only for certain types of scaling sequences which do not cause the estimation process to "forget" too quickly. This is not generally the case with the

[7] Subscripts "d" are omitted here since their algorithm does not involve this form of adaptation.

Dasgupta–Huang strategy.

Before proceeding, another comparison to the Dasgupta–Huang OBE algorithm should be made. One of the principal advantages of their method is the ability to conveniently prove convergence of the ellipsoid to a point ($\boldsymbol{\theta}_*$). The original Fogel and Huang paper [6] is sometimes cited as proving that the bounding ellipsoid in the Fogel–Huang OBE algorithm converges to a point under ordinary conditions on $\varepsilon_*(\cdot)$. In fact, the paper only proves this convergence for the case of unity weights so that the fundamental optimization process is not taken into account. No known proof of this desirable result for the Fogel–Huang OBE algorithm, or for any version of SM-WRLS exists, whether optimal or suboptimal checking is used. While the estimate itself is guaranteed to converge asymptotically under proper conditions on $\varepsilon_*(\cdot)$ (e.g. [10]), the ellipsoid is not guaranteed to diminish asymptotically. However, we have found empirically that the optimal and suboptimal tests tend to produce an ellipsoid with a similar "size" at a given point in the signal, and to produce similar estimates, in spite of the fact that the suboptimal test tends to use fewer data (see simulations below).

A further interpretation of (25) is possible which also allows the extension of the test to include "trace" minimization as well. A simple rationale for the suboptimal test is as follows:

Proposition 4 *If the test of (25) is met, then a positive optimal weight exists for either the volume or trace criterion.*

Proof: We show that the zero order cofficients a_0 and b_0, of (18) and (19), respectively, will never be positive if the test is met. Consider $a_0 = m\left[\gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1))\right] - \kappa_d(n-1)G_d(n)$. Write (11) for the downdated case at time $n-1$, then multiply through by $\kappa_d(n-1)$. The result is $[\boldsymbol{\theta} - \boldsymbol{\theta}_d(n-1)]^T \boldsymbol{C}_d(n-1)[\boldsymbol{\theta} - \boldsymbol{\theta}_d(n-1)] < \kappa_d(n-1)$. If $\boldsymbol{C}_d(n-1)$ is positive definite, this implies that $\kappa_d(n-1) > 0$. Further $G_d(n) = \boldsymbol{x}^T(n)\boldsymbol{C}_d(n-1)\boldsymbol{x}(n) > 0$, so $a_0 < 0$ if the test (25) is met. Now, consider $b_0 = \gamma(n) - \varepsilon^2(n, \boldsymbol{\theta}_d(n-1)) - H_d(n)I_d(n-1)\kappa_d(n-1)$. By similarly showing that $H_d(n) > 0$ and $I_d(n-1) > 0$, the desired result for the trace criterion is obtained. □

In the volume case, for example, the suboptimal check tests whether a_0 is negative *if the term* $-\kappa_d(n-1)G_d(n)$ *is neglected.* This ignored term is always negative and becomes small as n increases. For a given set of preceding optimal weights, $\lambda^*(1), \ldots, \lambda^*(n-1)$, *the optimal test will always accept a data set which is accepted by the suboptimal test.* While the converse is not true, the tests become similar for large n, and empirical evidence (see below) suggests that those data "missed" by the suboptimal test are not essential to good performance. A similar analysis applies to the coefficient b_0 of the trace algorithm.

With the inexpensive test afforded by (25), the checking procedure becomes an $\mathcal{O}(m)$ procedure.

Consequently, for sufficiently small ρ, the SM-WRLS algorithm can be run in $\mathcal{O}(m)$ time per n.

3.3 Simulation Studies and Further Discussion

OBE algorithms which do not include explicit adaptation measures have been demonstrated in numerous papers cited above. Our principal objective here is to briefly illustrate the use of the adaptive and, particularly, the $\mathcal{O}(m)$ suboptimal checking procedures.

We consider the identification of a time varying *AR(14)* model of the form

$$y(n) = \sum_{i=1}^{14} a_{i*}(n)y(n-i) + \varepsilon_*(n). \tag{29}$$

A set of "true" AR parameters were derived using linear prediction analysis (e.g. [36]) of order 14 on an utterance of the word "seven" by an adult male speaker. The original speech waveform is shown in Fig. 1 to illustrate the time varying nature of the signal. A 7000 point sequence, $y(n)$, was generated by driving the derived set of parameters with an uncorrelated sequence, $\varepsilon_*(n)$, which was uniformly distributed on $[-1, 1]$.

The speech signal was not used directly in this study for a simple reason. The problem of determining proper bounds for the model error is a nontrivial one for real speech, and a proper description of this point would seriously sidetrack the present discussion. Similarly, space would not permit a careful discussion of the performance of the algorithm in cases in which noise bounds are uncertain or violated. The predecessor (optimal, nonadaptive case) methods to those illustrated here have been successfully applied to real speech and these results are reported in [26] where some of these more difficult issues are also addressed. In the same vein, the artificial noise permits carefully controlled statistical properties. The model noise used here is uncorrelated, and this algorithm in its present form will converge to a bias if this is not the case. A discussion of colored noise, while interesting and useful, is beyond the scope of this paper. The interested reader is referred to [24],[28],[34]. While the uniform distribution chosen here has become conventional in testing OBE algorithms, it is worth noting that the performance of the methods is bound to be affected to some extent by the choice of this distribution. This becomes clear upon recognizing that the algorithm tends to favor the acceptance of data at time n when the residual is large. In some preliminary runs with bounded but nonuniform distributions, we do not find these effects to be very significant.

In the simulations below, we apply the conventional and adaptive SM-WRLS algorithms with "volume" optimization to the identification of the a_{i*} parameters. We discuss several simulation results. Only the result for a_{4*} is shown in each case to conserve space. Of the 14 parameters, a_{4*} emerged as the most difficult to track and gave the *worst* performance. Each figure shows two

curves, one for the true parameter, the other for the estimate obtained by the algorithm under study.

We have noted above that that OBE algorithms often exhibit good tracking capabilities by virtue of their optimal data weighting strategies, even when not explicitly designed to be adaptive (e.g. [24]–[29]). However, the tracking ability of "nonadaptive" OBE algorithms is somewhat unpredictable and fortuitous, especially for fast time variations. Further they are subject to divergence if the true parameters move outside the feasible set. Nevertheless, SM-WRLS and other OBE algorithms often demonstrate this inherent ability. The present example is contrary. Figure 2 illustrates the result of applying SM-WRLS to the time varying waveform. The estimate clearly fails to appropriately track the true parameter in this case.

Before proceeding, let us use the present result to emphasize a principal point made in the paper. The result of Fig. 2 is achieved using only the fraction $\rho = 0.079$ of the data. Other examples are found in the literature where *good* tracking is achieved with similar, or even smaller, fractions of the data used. It is important to keep in mind, however, that the computational complexity of the SM-WRLS algorithm is only a factor of about five better than conventional RLS, and the "$\rho = 0.079$" figure must not be interpreted to the contrary. Herein lies the motivation for the suboptimal checking procedure.

Next, we show the simulation results of the variations on the adaptive SM-WRLS algorithm. Figure 3 shows the results of the windowed SM-WRLS algorithm using windows of lengths 500, 1000, and 1500. This strategy uses the fractions $\rho = 0.221, 0.174$, and 0.143 of the data, respectively, but remains an $\mathcal{O}(m^2)$ process because optimal checking is used. Additionally, each time an *accepted* point occurs at the trailing edge of the window, a back-rotation is needed to effect adaptation. This implies an average number of back-rotations $b \approx \rho$ per n (see Section 3.1). More data and more rotations than with the unmodified SM-WRLS algorithm are used, but more accurate estimates result and the time varying parameters are tracked more quickly and accurately. As expected, adapting over smaller windows tended to improve time resolution, but increased the variation of the estimate and increased the number of points accepted. Conversely, the longer windows yielded smoother estimates using fewer data, but at the expense of slower tracking. While no window length in this range yielded grossly unacceptable estimates, the 1000 point window illustrated represents a good tradeoff between the demands of time and frequency resolution.

Figure 4 illustrates the use of suboptimal checking in conjunction with windowed SM-WRLS with a window of length 1000. Interestingly, the fraction of the data used is $\rho' = 0.087$ which is about half that required in the same experiment with optimal data checking (Fig. 3(b)). This means that the suboptimal checking not only reduced the computational effort of checking, but

also decreased by a factor of two the number of m^2 complexity rotations required. Nevertheless, the estimate trace is quite similar to the optimal case, the only difference being a slight increase in the variance near the end of the trace. Similar results were obtained for windows of length 500 and 1500.

The selective forgetting strategy chooses data sets to be removed from the system based on user defined criteria. Here the set of times to be back-rotated is as follows. Let $t' < n$ correspond to the "oldest" data set remaining in the estimate. Then $\mathcal{T}_{n-1} = \{t', \ldots, t''\}$, where the elements in the set are ordered, $t' < \cdots < t''$, and $t'' < n$ is the smallest time for which some other criterion is met. The determination of when to apply the forgetting procedure and when to stop removing data sets at a given time is discussed in the following.

The parameter a_{4*} to be tracked in this study is characterized by relatively fast time variations in the time region 2000 – 6000. The fact that the parameters change relatively slowly in the first 2000 points induces the algorithm to accept some points which, in turn, causes the ellipsoid volume to decrease. An increase in the "confidence" of the estimate results. Near time 2000, the ellipsoid volume becomes very small. When the parameters move rapidly away from their current location, they eventually move outside the ellipsoid which is therefore no longer a valid bounding ellipsoid. When this condition happens, it eventually leads to a negative value of $\kappa(n)$. For a stationary system, $\kappa(n)$ is always positive, so that this condition indicates that a violation of the theory (in particular, the violation of the assumption of stationary dynamics) has taken place[8]. A similar condition was also reported by Dasgupta and Huang [27] while applying their algorithm to nonstationary systems. In our simulation studies, we find that a negative $\kappa(n)$ is often an effective indicator of need for adaptation, and we use this criterion as the prompt to begin selective forgetting. Whenever accepting a data set causes $\kappa(n)$ to become negative, the algorithm starts rotating out the selected data sets until $\kappa(n)$ becomes positive again.

Figure 5 shows the simulation results of the selective forgetting strategy described here. The fraction $\rho = 0.129$ of the data is accepted by the estimation procedure and about 73% of these are back-rotated for adaptation. This implies a small "b" factor of about 0.094 per n so that adaptation is not expensive in this case. The checking process is still $\mathcal{O}(m^2)$, however, so the overall process remains of $\mathcal{O}(m^2)$ complexity. Suboptimal checking for the same experiment is illustrated in Fig. 6. In this case $\rho' = 0.088$ of the data is used with similar results. About 63% of these data are back-rotated, so that $b = 0.055$. Once again, the suboptimal test has preserved the quality of the estimate and lowered not only the checking complexity, but also the number of actual rotations that need be implemented.

[8] Mathematically, $\kappa(n) < 0$ indicates an ellipsoid of negative dimensions.

Compared to the windowed adaptive strategies, for this example the selective forgetting strategy yields smoother estimates using even fewer computations, but with poorer time resolution. (Recall that a_{4*} was found to be the most difficult to track in this simulation, so that this result is the worst case.) In general, we have found that selective forgetting (as employed here) generally uses fewer data and produces smoother estimates, but the tracking ability is not as reliable (though sometimes superior) to the windowed method [16],[34]. In fact, the selective forgetting strategy (as used here) tends to outperform windowing in cases of *very* fast time variations in the dynamics. The conservative schedule of back-rotations employed in the present technique accounts for this observation. $\kappa(n) > 0$ is only a necessary condition for the true parameters to be inside the current ellipsoid. The fact that $\kappa(\cdot)$ goes negative at a particular time does not precisely determine the point at which system dynamics began to change. If the variations are slow, this may occur (if at all) long after the dynamics begin to change. In fact, $\kappa(n) < 0$ often indicates a rather severe breakdown of the process indicating that the "true" parameters have moved well outside the current ellipsoid at time n. In cases of *fast* changing dynamics this "breakdown" occurs rapidly enough to render the condition "$\kappa(n) < 0$" a good locator of changing dynamics which require "immediate" adaptation to preserve the integrity of the process. The present example represents a very challenging case in the sense that variations apparently occur too rapidly to be tracked by standard SM-WRLS (see Fig. 2), yet not quickly enough to allow very high time resolution by the chosen selective forgetting method. Other methods for selection leading to a more aggressive elimination of past data may assist in the tracking at the expense of higher fractions of data used.

4 Architectural Solutions to Achieving $\mathcal{O}(m)$ Time

4.1 Systolic Architecture

In this section we develop parallel architectures on which both suboptimal and optimal checking versions of SM-WRLS will run in $\mathcal{O}(m)$ time. Here the efficiency is achieved by parallelism so that the number of operations is *effectively* reduced by simultaneous execution of many computations.

In the following we will assume the use of SM-WRLS (no scaling) for simplicity. Unlike the sequential case, however, scaling can be done in parallel here and does not add a significant computational burden. The modification of the following to include scaling is straightforward. We also use ellipsoid volume minimization for optimization, but a similar machine may be developed to implement trace optimization.

We first discuss the "nonadaptive" case. The fundamental parallel solution is made possible by the QR-WRLS version of SM-WRLS. The main computational requirements are a GR processor

(to effectively execute the QR decomposition) to update the matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$ at each step, and a back substitution (BS) processor to solve for the scalar $G(n)$ and also for the estimate $\boldsymbol{\theta}(n)$ at each n. Systolic processors for these operations, based on the original work of Gentleman and Kung [37] and Kung and Leiserson [38], are well known. It is the purpose of this section to manifest this algorithm as a parallel architecture based on these processors.

The need for implementing the algorithm on a parallel architecture arises from the fact that portions of the algorithm are compute-bound, specifically, updating the matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$ and computing the value $G(n)$ and the parameter vector $\boldsymbol{\theta}(n)$. The architecture that speeds up the computation of these quantities and satisfies the desirable characteristics of systolic arrays (SA's) is shown in Fig. 7. This architecture provides an improvement over that described in [39] by replacing the global buses with local buses for communication between adjacent cells. *For simplicity of notation, the figure shows a purely autoregressive case of order three, AR(3). Once the processor is understood, it should be clear that the architecture is perfectly capable of handling the general LP model case discussed above.* In the discussion below, the vector notations $\boldsymbol{x}(n)$ and $\boldsymbol{\theta}(n)$ are used, however, the architecture of Fig. 7 uses the vectors $\boldsymbol{y}(n)$ and $\boldsymbol{a}(n)$ instead to denote the special case AR(3), where $\boldsymbol{y}(n) = [y(n-1)\ y(n-2)\ y(n-3)]^T$.

The architecture is composed of two SA's, several memory management units (i.e., *First-in First-out* (FIFO) and *Last-in First-out* (LIFO) stacks[9]), multiply-add units (MAU's), multiplexers (MPX's), and demultiplexers (DMX's). The first SA is a *triangular* array that performs QR decomposition using GR's [37, 40] which are particularly suitable for solving recursive linear LSE problems. The diagonal (circular) cells perform the "Givens generation" (GG) operations and all other (square) cells in the triangular array perform the GR operations. There is a delay element at the lower right-hand corner of the triangular array that is used to synchronize the flow of the generated entries into the FIFO stacks and to simplify the control of these stacks once they are filled and ready to output their contents to the BS array. The operations performed by this array are shown in Fig. 8 [37, 40]. Therefore, the triangular array rotates the new data set into the upper triangular matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$, where the t_{ij} cells update the matrix $\boldsymbol{T}(n)$ and the right-hand column (d_{1j}) cells update the vector $\boldsymbol{d}_1(n)$. The element t_{ij} denotes the i, j^{th} element of the matrix $\boldsymbol{T}(n)$ and the element d_{1j} denotes the j^{th} element of the vector $\boldsymbol{d}_1(n)$.

The second array is a linear array that performs the BS operations shown in Fig. 9 [38]. Note that the same BS array is used to solve for the vectors $\boldsymbol{g}(n+1)$ and $\boldsymbol{\theta}(n)$ with the data provided to

[9]The architecture shown in Fig. 7 does not include any of the LIFO stacks that were used to hold the matrix $T(n)$ in the architecture reported in [39]. This is achieved by slightly increasing the complexity of the cells used in the triangular array so that they can be used as storage elements as well. This is facilitated by the diagonal interconnections between adjacent cells which now constitute the LIFO stacks.

the appropriate cells in the required order by the FIFO and LIFO stacks. The FIFO stacks feed the lower triangular matrix $\boldsymbol{T}^T(n)$ to solve for the vector $\boldsymbol{g}(n+1)$, and hence, the value $G(n+1)$. The LIFO stacks feed the upper triangular matrix $\boldsymbol{T}(n)$ to solve for the parameter vector $\boldsymbol{\theta}(n)$. The values $G(n+1) = \| \boldsymbol{g}(n+1) \|^2$ and $\| \boldsymbol{d}_1(n) \|^2$ are generated by the MAU's shown in Fig. 10. The number of segments in each stack is equal to the number of elements the stack holds. Therefore, the leftmost stack consists of m segments, whereas the rightmost stack has only one segment.

The system shown in Fig. 7 works as follows. The first m+1 data sets (with appropriate weights) enter the triangular array (from the top) in a skewed order, and the matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$ is generated and stored inside the cells. A shift register with appropriate feedback connection and data sequencing can be used to hold and feed the data set to the array. The initial upper triangular matrix residing in the array, and corresponding to the first $m+1$ data sets, is ready after $3m+1$ GG time cycles. The GG time cycle is that of the triangular array performing the GG operations without square roots, which is the time required to perform five flops [40],[41]. In order to prevent data collision, the flow of data in the triangular array moves along a corresponding wavefront and is controlled by the slowest cells in the array, *viz.*, GG cells. The data are fed to the array one (skewed) data set at a time, therefore, the contents of each cell remains constant after the completion of the current recursion. After the new data set is rotated into the matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$, the vectors $\boldsymbol{g}(n+1)$ and $\boldsymbol{\theta}(n)$ are computed. All the t_{ij} cells in the triangular array load their contents on the t_{out} lines ($t_{out} \leftarrow x$), and then pass these elements across the diagonal lines ($t_{out} \leftarrow t_{in}$) (see Figs. 7 and 8). This obviates LIFO stacks. The FIFO stacks are still needed, however, to compute the vector $\boldsymbol{g}(n+1)$. The FIFO stacks are filled with the elements of the lower triangular matrix $\boldsymbol{T}^T(n)$ as they are generated. This is done by loading the t_{ij} entry on the t_{out} line ($t_{out} \leftarrow x$) when it is generated. This entry propagates down the diagonal cells (with the function $t_{out} \leftarrow t_{in}$) until it arrives at and fills the appropriate FIFO stack. For the cells in the right-hand column, which generate the vector $\boldsymbol{d}_1(n)$, the operations are different because it is this column that constitutes the LIFO stack for the vector $\boldsymbol{d}_1(n)$. Hence, after the new data set is rotated into the array, all the cells in the right-hand column load their contents on the x_{out} lines ($x_{out} \leftarrow x$), and then they pass these elements down the column ($x_{out} \leftarrow x_{in}$) (see Figs. 7 and 8). The output x_{out} leaving the bottom cell in this column passes through the delay element and is routed to both the MAU and the MPX feeding the d_{1j} elements to the BS array. The elements d_{1m} and t_{mm} leave the triangular array at the same time because of this delay element. The timing diagram of the triangular array is shown in Table 1. In this table, the inputs refer to the elements fed to the cells in the top row. The circle ($\bigcirc$) represents the GG cell and the square ($\square$) represents the GR cell (see Fig. 7). The outputs refer to the elements that are produced in the array cells and are written columnwise; i.e.,

the first column in the table represents the first column in the array, and so on.

The BS array is used to solve for the vectors $\boldsymbol{g}(n+1)$ and $\boldsymbol{\theta}(n)$. The vector $\boldsymbol{g}(n+1)$ is solved using (20) and the parameter vector $\boldsymbol{\theta}(n)$ using (6). Therefore, the vector $\boldsymbol{g}(n+1)$ is generated from the matrix $\boldsymbol{T}^T(n)$, which is residing in the FIFO stacks, and the vector $\boldsymbol{x}(n+1)$ which is available. The entries are fed to the BS array every other BS time cycle, where the BS time cycle is the time required to perform one flop. As the g_i entries are output from the left-end processor of the BS array, they enter the MAU to generate the value $G(n+1)$ after $2m+1$ BS time cycles. Likewise, the parameter vector $\boldsymbol{\theta}(n)$ is generated using the matrix $\boldsymbol{T}(n)$ and the vector $\boldsymbol{d}_1(n)$ which are stored in the triangular array. Starting one BS time cycle after the initiation of the first BS operation, the appropriate entries (of the second BS operation) are also fed to the BS array every other BS time cycle. The parameter vector $\boldsymbol{\theta}(n)$ is output from the left-end processor of the BS array in reversed order and interleaved with the vector $\boldsymbol{g}(n+1)$ as shown in Fig. 7. The value $\| \boldsymbol{d}_1(n) \|_2^2$ is generated using a MAU one BS time cycle after the last (m^{th}) element of the vector $\boldsymbol{d}_1(n)$ is generated. The timing diagram of the BS array is shown in Table 2 in which the inputs refer to the elements fed to the shown cells, and the outputs refer to the elements produced by the left-end processor in the array.

The values $\kappa(n)$ and $\epsilon^2(n+1, \boldsymbol{\theta}(n))$ are then computed, and hence, the value $\lambda_{n+1}^{*}(n+1)$ which determines whether the new data set is to be accepted or not. If the new data set is accepted, then the *weighted* new data set enters the triangular array and the same procedure described above takes place producing a new $[\boldsymbol{T}(n+1) \mid \boldsymbol{d}_1(n+1)]$ matrix after $2m+1$ GG time cycles, and therefore, an updated $G(n+2)$, $\boldsymbol{\theta}(n+1)$, and $\kappa(n+1)$. On the other hand, if the new data set is rejected, then the triangular array preserves its contents (hold state), but the value $G(n+2)$ is updated to make the decision concerning the next data set. In the latter case, the same $\boldsymbol{T}^T(n+1)$ matrix is used as the previous $\boldsymbol{T}^T(n)$ matrix, and hence, the feedback on the FIFO stacks. This procedure is repeated for every new data set.

The computational complexities (in flops per data set) for the architecture of Fig. 7 is approximated by [16]

$$f_{\substack{opt \\ parallel}} \sim \mathcal{O}(3m) + \rho \mathcal{O}(11m) \tag{30}$$

where the first term accounts for checking and the second for solution update, with ρ defined as usual. As noted at the outset, the complexities of the solution are *parallel* complexities in the sense that they denote the *effective* number of operations per data set, though many processors can be performing this number of operations simultaneously. Accordingly, the parallel complexity indicates the time it takes the parallel architecture to process the data, regardless of the total number of operations performed by the individual cells. The GG and GR operations constitute the

main computational load of the algorithm as shown in Table 3. In this table, the number of flops associated with the GR's is multiplied by five to account for the GG cycle time. These operations are avoided when the data set is rejected, and thus, a significant savings in computation time is achieved.

Suboptimal checking may also be used in conjunction with the parallel processing. In this case it is simply unnecessary for the processor to compute the second, third, and fourth items in Table 3 in order to check the incoming data set. The reduction in computation, which is is not as significant as in the sequential processing case, is reflected by the approximation

$$f_{\substack{subopt \\ parallel}} \sim \mathcal{O}(m) + \rho' \mathcal{O}(11m) \tag{31}$$

for small ρ' [16].

4.2 Adaptive Compact Architecture

The architecture described above can be modified to improve cell utilization and to incorporate adaptation by back-rotation. The basic idea behind the compact architecture is to map the triangular array of Fig. 7 into a linear array (called the GR array), that is, mapping all of the GG cells into one GG cell and all the GR cells that are on the same diagonal into one GR cell. This constitutes a permissible schedule because the projection vector, $\vec{d}$, is parallel to the schedule vector, $\vec{s}$, and all the dependency arcs flow in the *same* direction across the hyperplanes (e.g. [42, Ch. 3]). In other words, this schedule satisfies the conditions $\vec{s}^T \vec{d} > 0$ and $\vec{s}^T \vec{e} \geq 0$, for any dependence arc $\vec{e}$.

The compact architecture implementation of the adaptive SM-WRLS algorithm is shown in Fig. 11. The operations performed by this architecture are similar to those of Fig. 7 with the exception that the GG and GR cells are now capable of performing back rotation (see Fig. 12) and are embedded in a slightly more complicated modules needed for scheduling. These modules are called GG′ and GR′, and are shown in Fig. 13.

This architecture uses $\mathcal{O}(m)$ cells (one GG′ cell and m GR′ cells) compared with $\mathcal{O}(m^2)$ cells (m GG cells and $(m^2 + m)/2$ GR cells) used in the architecture shown in Fig. 7, and yet has the same computational efficiency per n. Note however that the LIFO stacks that were embedded in the triangular array of Fig. 7 are now needed to hold the matrix $\boldsymbol{T}(n)$.

The system shown in Fig. 11 works as follows. Each data set (with its optimal weight) enters the GR array (from the top) in a skewed order, and the matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$ is generated and stored in the appropriate memory units. Note that the GR array can operate in two modes, forward ($\delta = +1$) and backward ($\delta = -1$) rotation modes (see Fig. 12). In the backward rotation mode, the data set to be removed is re-introduced to the GR array with the appropriate weight. At the end of each

recursion, the FIFO stacks contain the lower triangular matrix $\boldsymbol{T}^T(n)$ needed to solve for the vector $\boldsymbol{g}(n+1)$, and hence, the value $G(n+1)$. The LIFO stacks contain the upper triangular matrix $\boldsymbol{T}(n)$ needed to solve for the parameter vector $\hat{\boldsymbol{\theta}}(n)$. The values $G(n+1) = \| \boldsymbol{g}(n+1) \|^2$ and $\| \boldsymbol{d}_1(n) \|^2$ are generated by the MAU's. Note that the values which were propagating downward in the triangular array of Fig. 7 are now propagating leftward due to the new scheduling. Note also that the vector $\boldsymbol{d}_1(n)$ is treated differently from the matrix $\boldsymbol{T}(n)$. When the element d_{1j} is computed, it is stored in an internal register in the GR' cell (see Fig. 13). After generating and storing the matrix $[\boldsymbol{T}(n) \mid \boldsymbol{d}_1(n)]$, the processor is ready to compute the vectors $\boldsymbol{g}(n+1)$ and $\boldsymbol{\theta}(n)$ using the BS array. The vector $\boldsymbol{d}_1(n)$ is downloaded into the latches which serve as a LIFO stack used in conjunction with the other LIFO stacks (containing the matrix $\boldsymbol{T}(n)$) to solve for the parameter vector $\boldsymbol{\theta}(n)$. The timing diagram of the GR array is shown in Table 4 in which the input (output) columns show the elements that are input (output) to (from) the corresponding GG ($\bigcirc$) or GR ($\square$) cells. Compared to the triangular array of Fig. 7, it is noted that the cell utilization per update (or downdate) has increased by a factor of 2.25 for the case when $m = 3$, or by $(.5m^2 + 1.5m)/(m+1)$ in general. The operations and timing diagram of the BS array are described in detail above.

The adaptive compact architecture of Fig. 11 has slightly more complicated cells than that of Fig. 7, but requires the same number of operations to check and incorporate a data set. However, the compact architecture processor may additionally be used to back-rotate a data set for adaptation. The forward and backward rotation modes have the same parallel complexity. Therefore, it is only necessary to add terms of the form $b\mathcal{O}(11m)$ to either (30) or (31) to account for back-rotation, where b has the usual meaning.

5 Conclusions

Two general contributions have been made to the theory and application of OBE algorithms for linear-in-parameters models. We have first suggested that all reported OBE algorithms, both nonadaptive and adaptive, can be placed into a general framework which is intimately related to recursive LSE processing. A flexible form of explicit adaptation has been demonstrated within this framework. In particular, a general technique based on "back-rotation" within the context of the QR-decomposition based version of WRLS offers a flexible array of adaptation strategies and good tracking ability. Secondly, two very different approaches to rendering a specific OBE algorithm, SM-WRLS, of $\mathcal{O}(m)$ per n computational complexity have been proposed. The computational complexity of the optimal OBE algorithms is of $\mathcal{O}(m^2)$ flops per n in spite of the highly discriminating

data selection through set-membership criteria. This fact has not been made clear in the literature. This paper has demonstrated both an algorithmic and an architectural solution to this problem, making the SM-WRLS method superior to many other LSE techniques in a computational sense. In signal processing applications, this computational advantage is complemented by the existence of the feasible set of solutions for which many other interesting purposes may be found.

References

[1] E. Walter (editor), *Mathematics and Computers in Simulation*, Special issue on parameter identifications with error bound, vol. 32, Dec. 1990.

[2] Cs. Bányász and L. Keviczky (editors), *Proceedings of the 9th IFAC / IFORS Symposium on Identification and System Parameter Estimation*, vols. 1 & 2, Budapest, July 1991.

[3] E. Walter and H. Piet-Lahanier, "Estimation of parameter bounds from bounded-error data: A survey," *Mathematics and Computers in Simulation*, vol. 32, pp. 449-468, 1990.

[4] M. Milanese and A. Vicino, "Estimation theory for dynamic systems with unknown but bounded uncertainty: An overview," *Proceedings of the 9th IFAC / IFORS Symposium on Identification and System Parameter Estimation*, vol. 2, pp. 859-867, Budapest, July 1991.

[5] J.R. Deller, Jr., "Set-membership identification in digital signal processing," *IEEE ASSP Magazine*, vol. 6, pp. 4-22, Oct. 1989.

[6] E. Fogel and Y.F. Huang, "On the value of information in system identification - Bounded noise case," *Automatica*, vol. 18, pp. 229-238, 1982.

[7] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Baltimore, MD: Johns-Hopkins Univ. Press, Ch. 6, 1983.

[8] J.P. Norton, *An Introduction to Identification*, London and Orlando, Florida: Academic Press, 1986.

[9] C.R. Johnson, *Lectures on Adaptive Parameter Estimation*, Englewood Cliffs, New Jersey: Prentice-Hall, 1988.

[10] D. Graupe, *Time Series Analysis, Identification, and Adaptive Systems*, Malabar, Florida: Krieger, Ch. 5, 1989.

[11] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, Cambridge, MA: MIT Press, pp. 14-15 & Sect. 2.2.1., 1983.

[12] T.C. Luk and J.R. Deller, Jr., "A nonclassical WRLS algorithm," *Proc. 23rd Ann. Allerton Conf.*, pp. 732-741, 1985.

[13] J.R. Deller, Jr. and D. Hsu, "An alternative adaptive sequential regression algorithm and its application to the recognition of cerebral palsy speech," IEEE Trans. Circ. and Syst., vol. CAS-34, pp. 782-787, July 1987.

[14] J.R. Deller, Jr. and G.P. Picaché, "Advantages of a Givens rotation approach to temporally recursive linear prediction analysis of speech," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 37, pp. 429-431, March 1989.

[15] J.R. Deller, Jr., "A 'systolic array' formulation of the optimal bounding ellipsoid algorithm," *IEEE Trans. Acoust., Speech, and Signal Process.*, vol. 37, pp. 1432-1436, Sept. 1989.

[16] S.F. Odeh, *Algorithms and Architectures for Adaptive Set Membership-based Signal Processing* (Ph.D. Dissertation), Michigan State University, East Lansing, 1990.

[17] S.F. Odeh and J.R. Deller, Jr., "Systolic algorithms for adaptive set-membership identification," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '90*, Albuquerque, vol. 5, pp. 2419-2422, Apr. 1990.

[18] S.H. Mo and J.P. Norton, "Fast and robust algorithm to compute exact polytope parameter bounds," *Mathematics and Computers in Simulation*, vol. 32, pp. 481-493, 1990.

[19] V. Broman and M.J. Shensa, "A compact algorithm for the intersection and approximation of N-dimensional polytopes," *Mathematics and Computers in Simulation*, vol. 32, pp. 469-480, 1990.

[20] E. Walter, and H. Piet-Lahanier, "Exact recursive polyhedral description of the feasible parameter set for bounded-error models," *IEEE Trans. Automatic Control*, vol. AC-34, pp. 911-915, 1989.

[21] M. Milanese and G. Belaforte, "Estimation theory and uncertainty intervals evaluation in the presence of unknown but bounded errors: Linear families of models and estimators," *IEEE Trans. Automatic Control*, vol. AC-27, pp. 408-414, 1982.

[22] G. Belforte, B. Bona, and V. Cerone, "Parameter estimation algorithms for a set-membership description of uncertainty," *Automatica*, vol. 26, pp. 887-898, 1990.

[23] A.K. Rao, Y.F. Huang, and S. Dasgupta, "ARMA parameter estimation using a novel recursive estimation algorithm with selective updating," *IEEE Trans. Acoust. Speech and Signal Process.*, 38, pp. 447-457, March 1990.

[24] A.K. Rao and Y.F. Huang, "Recent developments in optimal bounding ellipsoid parameter estimation, *Mathematics and Computers in Simulation*, vol. 32, pp. 515-526, Dec. 1990.

[25] L. Pronzato, E. Walter, H. Piet-Lahanier, "Mathematical equivalence of two ellipsoid algorithms for bounded error estimation," *Proc. 28th IEEE Conf. on Decision and Control*, Tampa, pp. 1952-1955, 1989.

[26] J.R. Deller, Jr. and T.C. Luk, "Linear prediction analysis of speech based on set-membership theory," *Computer Speech and Language*, vol. 3, pp. 301-327, Oct. 1989.

[27] S. Dasgupta and Y.F. Huang, "Asymptotically convergent modified recursive least squares with data dependent updating and forgetting factor for systems with bounded noise," *IEEE Trans. Information Theory*, vol. IT-33, pp. 383-392, 1987.

[28] J.P. Norton, "Identification of parameter bounds for ARMAX models from records with bounded noise," *Int. J. Control*, vol. 45, pp. 375-390, 1987.

[29] Y.F. Huang, "A recursive estimation algorithm using selective updating for spectral analysis and adaptive signal processing," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-34, pp. 1331-1334, 1986.

[30] J.P. Norton, "Recursive computation of inner bounds for the parameters of linear models," *Int. J. Control*, vol. 50, pp. 2423-2430, 1989.

[31] A. Vicino and M. Milanese, "Optimal approximation of feasible parameter set in estimation with unknown but bounded noise," *Proceedings of the 9th IFAC / IFORS Symposium on Identification and System Parameter Estimation*, vol. 2, pp. 1365-1369, Budapest, July 1991.

[32] J.R. Deller, Jr. and S.F. Odeh, "SM-WRLS algorithms with an efficient test for innovation," *Proceedings of the 9th IFAC / IFORS Symposium on Identification and System Parameter Estimation*, vol. 2, pp. 1044-1049, Budapest, July 1991.

[33] J.P. Norton and S.H. Mo, "Parameter bounding for time-varying systems," *Mathematics and Computers in Simulation*, vol. 32, pp. 527-534, 1990.

[34] J.R. Deller, Jr., S.F. Odeh, and M. Nayeri, "Set-membership algorithms for signal processing," *Proc. of the IEEE*, in review.

[35] L.E. Dickson, *A First Course in the Theory of Equations*, New York: Wiley, 1952.

[36] J. Makhoul, "Linear prediction: A tutorial review," *Proc. of the IEEE*, vol. 63, pp. 561-580, 1975.

[37] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays," *Proc. Soc. Photo-Optical Instrumentation Engineers: Real-Time Signal Processing IV*, vol. 298, pp. 19-26, 1981.

[38] H.T. Kung and C. Leiserson, "Algorithms for VLSI processor arrays," Rpt. No. MU-CS-79-103, Dept. of Computer Sci., Carnegie-Mellon Univ., 1978. Reprinted in C. Mead and L. Conway, *Introduction to VLSI Systems*, Reading, MA: Addison-Wesley, Section 8.3, 1980.

[39] J.R. Deller, Jr. and S.F. Odeh, "Implementing the optimal bounding ellipsoid algorithm on a fast processor," *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Process. '89*, Glasgow, vol. 2, pp. 1067-1070, May 1989.

[40] J.G. McWhirter, "Recursive least squares minimization using a systolic array," *Proc. Soc. Photo-Optical Instrumentation Engineers: Real-Time Signal Processing VI*, vol. 431, pp. 105-112, 1983.

[41] W.M. Gentleman, "Least squares computations by Givens transformations without square roots," *J. Inst. of Math. and its Appl.*, Vol. 12, pp. 329-336, 1973.

[42] S.Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice Hall, Ch. 3, 1988.

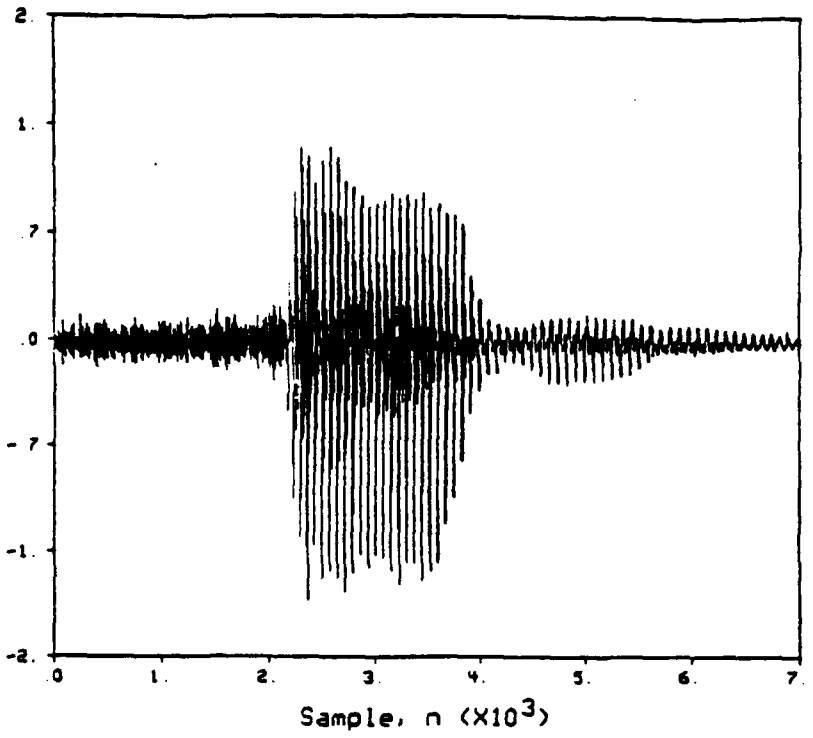


Figure 1: Acoustic waveform of the utterance "seven" upon which the time varying system in the simulation studies is based.

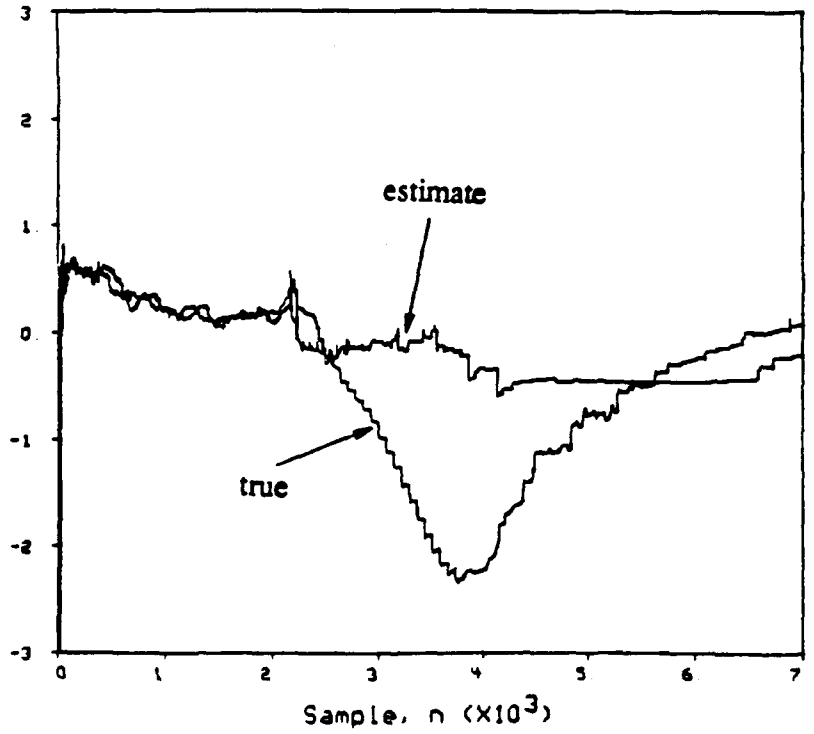
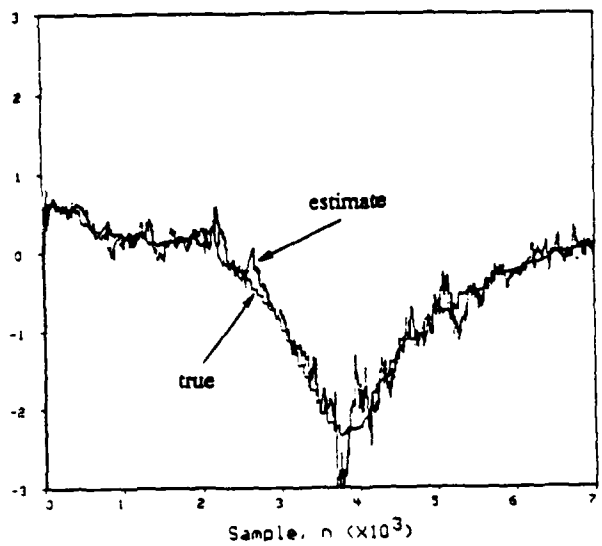
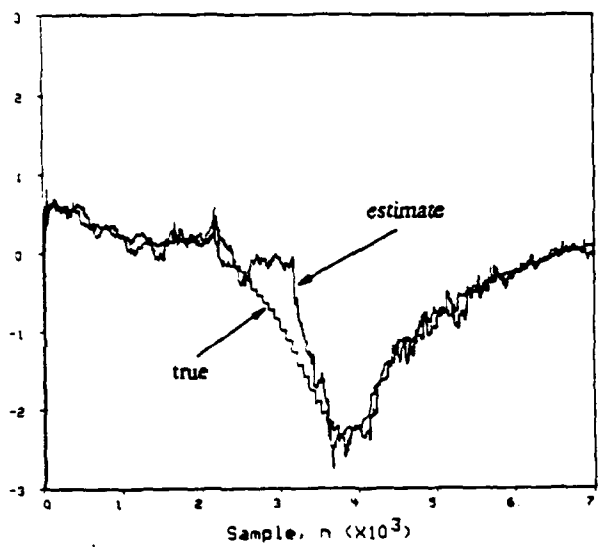


Figure 2: "Nonadaptive" SM-WRLS algorithm applied to the estimation of parameter a_{4*}. Only $\rho = 0.079$ of the data is used, but the estimate fails to track the true parameter.





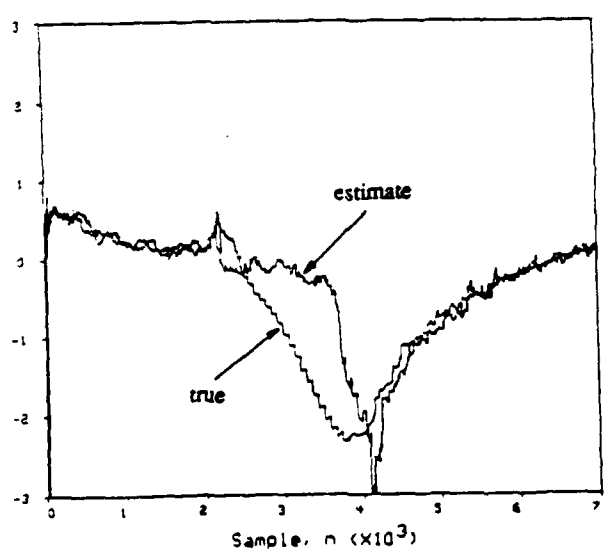


Figure 3: Windowed SM-WRLS with optimal data checking applied to the estimation of parameter a_{4o}. The window lengths are (a) 500, (b) 1000, and (c) 1500 points, and the fractions (a) $\rho = 0.221$, (b) 0.174, and (c) 0.143 of the data are used in the estimation.

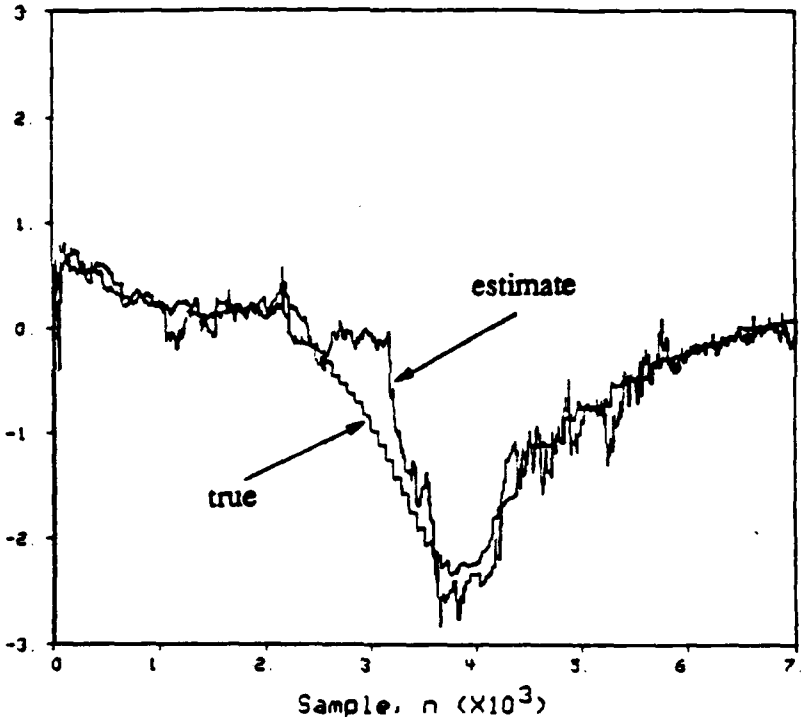


Figure 4: Windowed SM-WRLS with suboptimal data checking applied to the estimation of parameter a_{4*}. The window length is 1000 points and the fraction $\rho = 0.087$ of the data is used in the estimation.

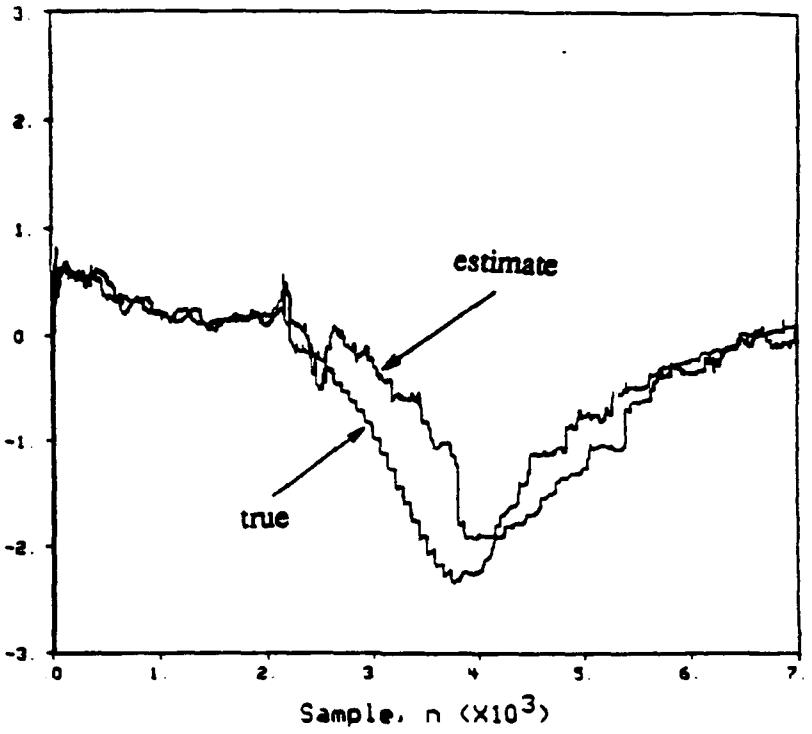


Figure 5: "Selective forgetting" SM-WRLS with optimal data checking applied to the estimation of the parameter a_{4*}. The criterion for selective removal of past points is described in the text. The fraction $\rho = 0.129$ of the data is used by the estimation procedure and the adaptation is computationally very inexpensive.

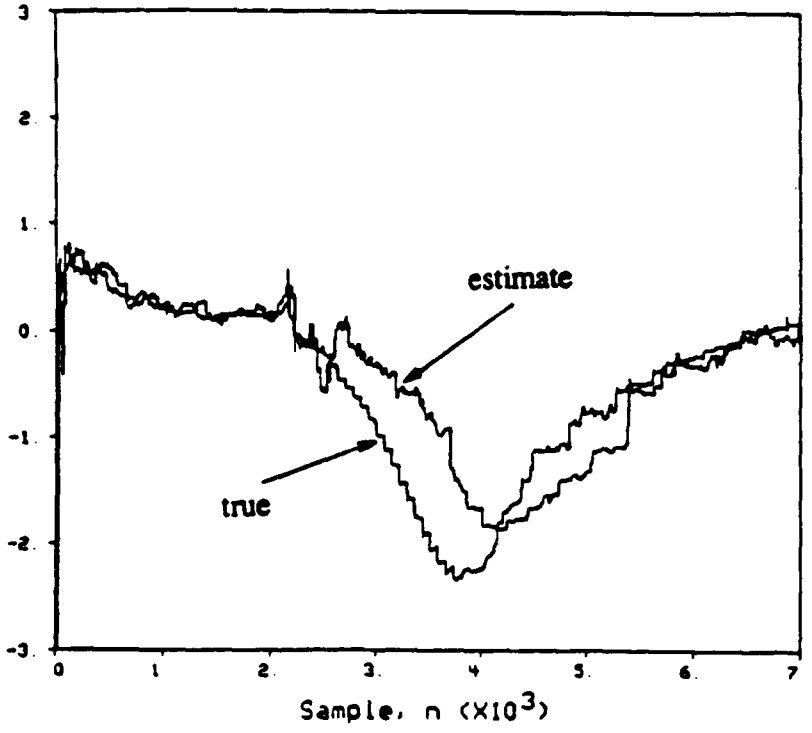


Figure 6: "Selective forgetting" SM-WRLS with suboptimal data checking applied to the estimation of the parameter a_{4*}. The criterion for selective removal of past points is described in the text. The fraction $\rho = 0.088$ of the data is used by the estimation procedure and the adaptation is computationally very inexpensive.

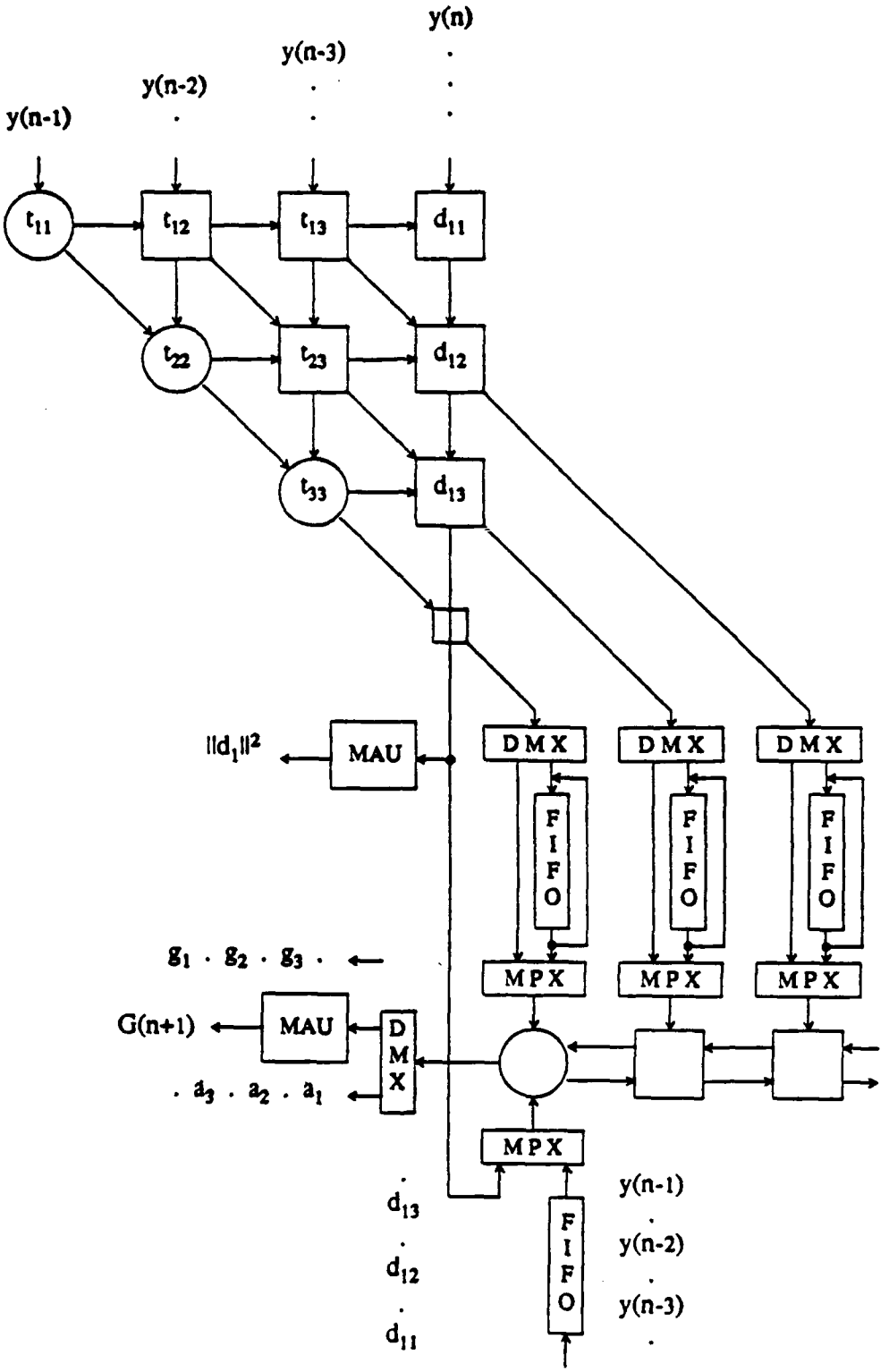
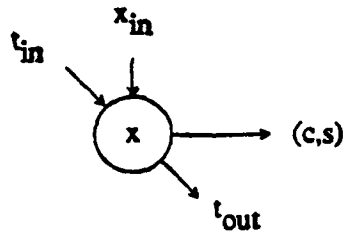
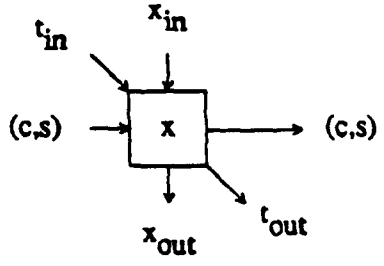


Figure 7: Systolic array implementation of the QR-WRLS based SM-WRLS algorithm. For simplicity, but without loss of generality, a pure autoregressive case of order three (AR(3)) is illustrated.



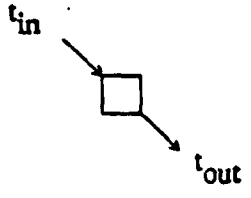
(a)

If (x_{in} = 0){
  c = 1
  s = 0
  t_{out} = t_{in}
}
else{
  temp = $[x^2 + (x_{in})^2]^{1/2}$
  c = x / temp
  s = x_{in} / temp
  x = temp
  t_{out} = x
}



(b)

If (x_{in} = 0 & c = 1 & s = 0)
  t_{out} = t_{in}
else{
  x = c x + s x_{in}
  x_{out} = -s x + c x_{in}
  t_{out} = x
}

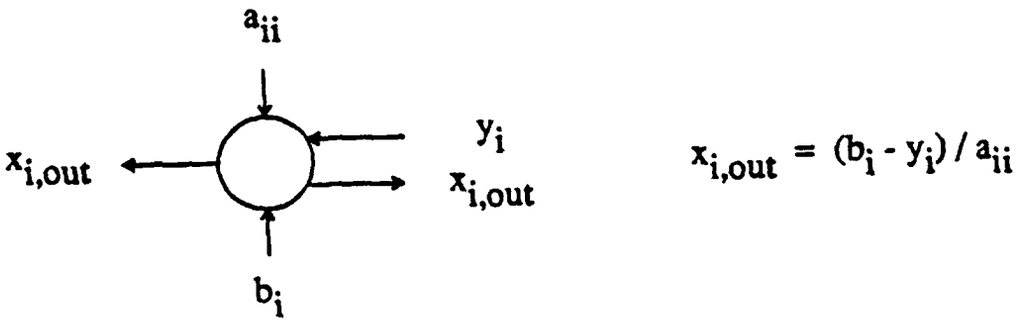


(c)

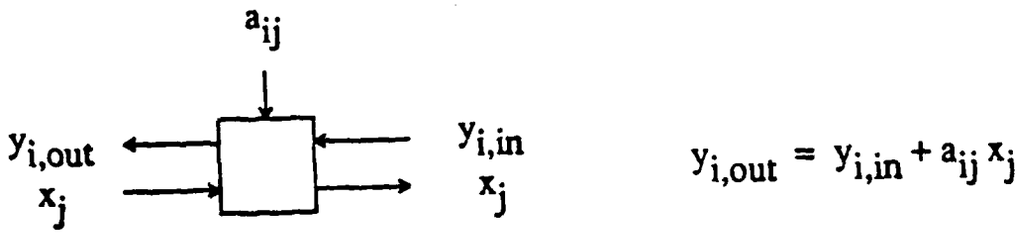
t_{out} = t_{in}

When the array is used as LIFO stacks: First cycle: t_{out} = x (t_{ij} cells)
x_{out} = x (d_1 cells)
All following cycles: t_{out} = t_{in} (t_{ij} cells)
x_{out} = x_{in} (d_1 cells)

Figure 8: The operations performed by the cells used in the triangular array of Fig. 7. (a) The Givens generation (GG) cells, (b) the Givens rotation (GR) cells, and (c) the delay element.



(a)



(b)

Figure 9: Operations performed by the back substitution array. (a) The left-end processor and (b) the multiply-add units. The initial $y_{i,in}$ entering the rightmost cell is set to 0.

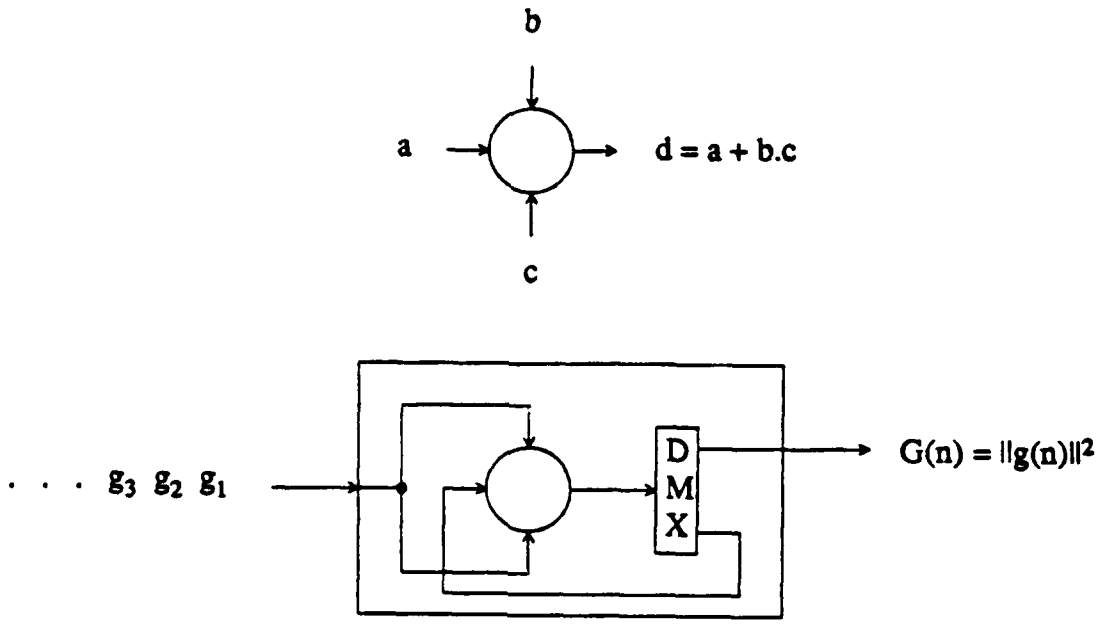


Figure 10: Multiply-add unit used in Fig. 7.

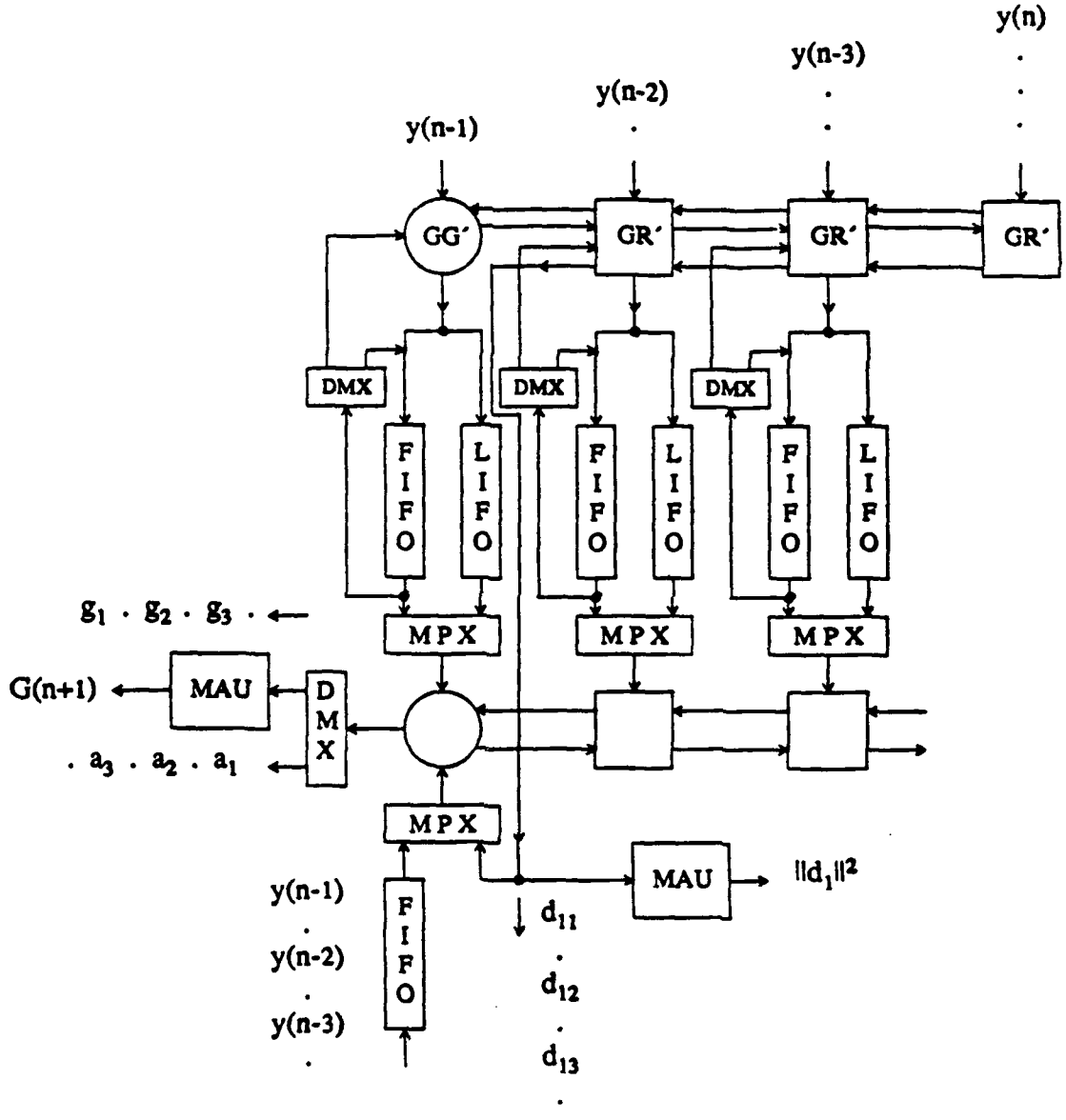
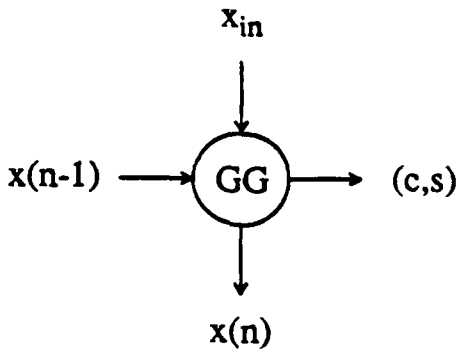
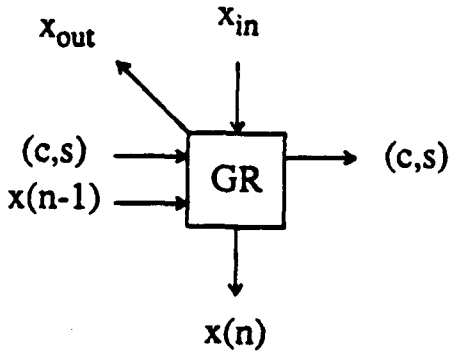


Figure 11: A compact architecture implementation of the adaptive SM-WRLS algorithm.



If ($x_{in} = 0$){
  c = 1
  s = 0
}
else{
  $x(n) = [x(n-1)^2 + \delta(x_{in})^2]^{1/2}$
  c = x(n-1) / x(n)
  $s = x_{in} / x(n)$
}

(a)



$x(n) = c\,x(n-1) + s\,x_{in}\delta$

$x_{out} = -s\,x(n-1)\delta + c\,x_{in}\delta$

(b)

Figure 12: Operations performed by (a) the GG and (b) the GR cells used in the modules of Fig. 11. $\delta = +1\ (-1)$ for rotating the data set into (out of) the system.

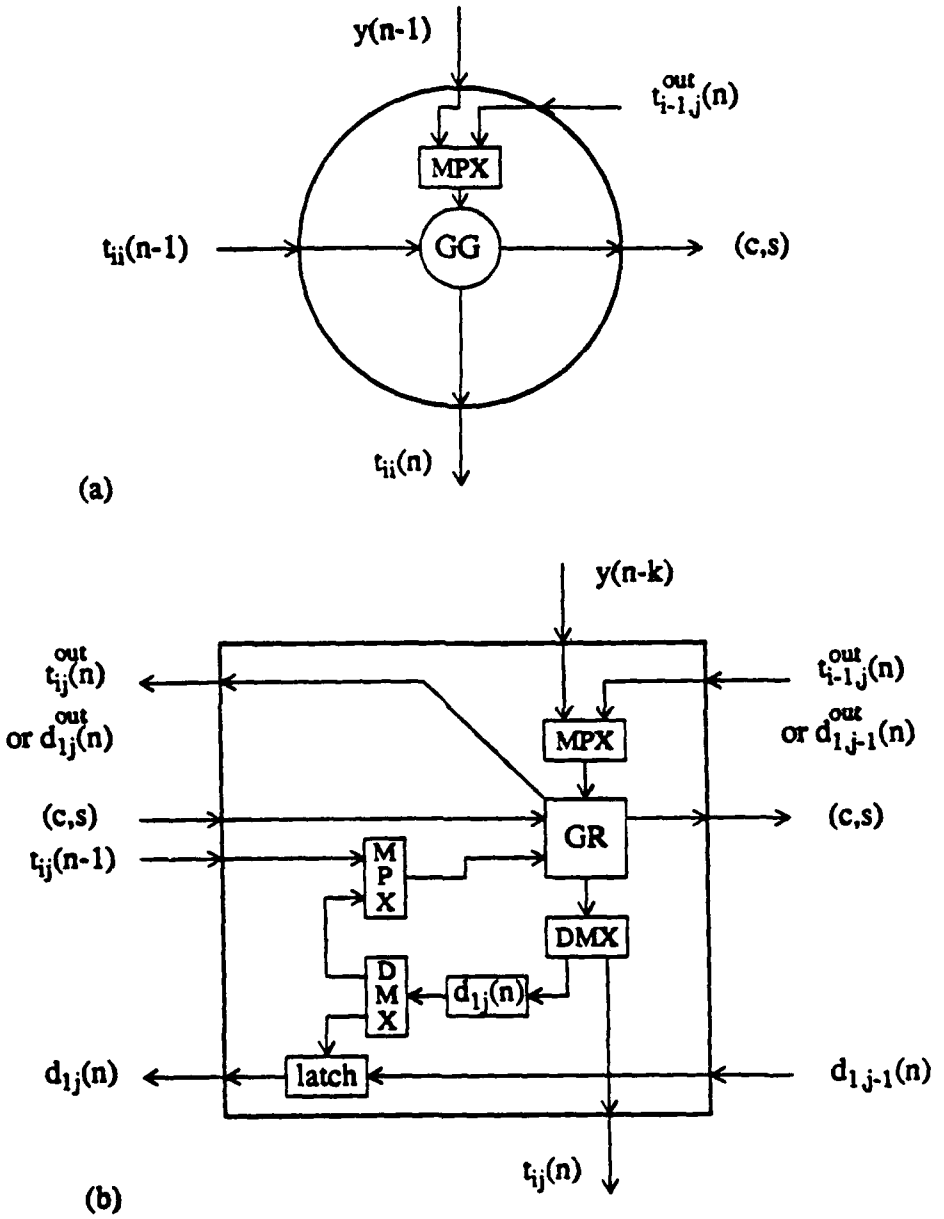


Figure 13: (a) GG′ module and (b) GR′ module used in the architecture of Fig. 11.

Time	Inputs ○	□	□	□	Outputs $[\mathbf{T}(n) \mid \boldsymbol{d}_1]$		
0	$y(n-1)$						
1		$y(n-2)$			t_{11}		
2			$y(n-3)$			t_{12}	
3				$y(n)$		t_{22}	t_{13}
4							t_{23} d_{11}
5							t_{33} d_{12}
6							d_{13}

Table 1: Timing diagram of the triangular array of Fig. 7.

Time	Inputs ○	□	□	Outputs ○
0	$t_{11}, y(n-1)$			
1	t_{33}, d_{13}	t_{12}		g_1
2	$t_{22}, y(n-2)$	t_{23}	t_{13}	$\hat{a}_3$
3	t_{22}, d_{12}	t_{23}	t_{13}	g_2
4	$t_{33}, y(n-3)$	t_{12}		$\hat{a}_2$
5	t_{11}, d_{11}			g_3
6				$\hat{a}_1$

Table 2: Timing diagram of the back substitution array of Fig. 7.

Element Computed	flops per n
$\varepsilon(n+1, \boldsymbol{\theta}(n))$	$m+1$
coefficient of quadratic (18)	7
$\lambda_{n+1}^{*}(n+1)$	$5+\sqrt{}$
$G(n+1)$ and $\boldsymbol{\theta}(n)$	$2m+1$
If data set is accepted:	
update	$m+1+\sqrt{}$
Givens rotations	$5(2m+1)$
$\kappa(n)$	4

Table 3: Numbers of operations required by the GG and GR cells in the architecture of Fig. 7.

Time	Inputs ○	□	□	□	Outputs ○	□	□	□
0	$y(n-1)$							
1		$y(n-2)$			t_{11}			
2			$y(n-3)$			t_{12}		
3				$y(n)$	t_{22}		t_{13}	
4						t_{23}		d_{11}
5					t_{33}		d_{12}	
6						d_{13}		

Table 4: Timing diagram of the GR array of the compact architecture of Fig. 11.